КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки
на тему:

ВЕБЗАСТОСУНОК ДЛЯ СТЕЙКІНГУ КРИПТОВАЛЮТ

Виконала студентка 4-го курсу
Олександра ТІМОФЄЄВА

(підпис)

Науковий керівник:
доктор фіз.-мат. наук, професор
Степан ШКІЛЬНЯК

(підпис)

Засвідчую, що в цій
роботі немає запозичень
з праць інших авторів без
відповідних посилань

Студент

(підпис)

Роботу розглянуто і допущено до
захисту на засіданні кафедри теорії
та технології програмування
«05» червня 2023 р.,
протокол № 18
Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ – 2023

РЕФЕРАТ

Загальний обсяг роботи 60 сторінок, основний текст викладено на 54 сторінках, 29 рисунків, 25 джерел посилань, 3 додатки.

БЛОКЧЕЙН, СТЕЙКІНГ, КРИПТОВАЛЮТА, ВЕБЗАСТОСУНОК, МОДЕЛЮВАННЯ, АЛГОРИТМИ КОНСЕНСУСУ, СМАРТ-КОНТРАКТ, SOLIDITY.

Об’єктом дипломної роботи є блокчейн, криптовалюта та процеси стейкінгу криптовалют. Предметом роботи є вебзастосунок для стейкінгу криптовалют.

Метою роботи є розробка децентралізованого сервісу, що надає можливість користувачу отримувати стабільний пасивний дохід за зберігання коштів, а також надійне зберігання всієї актуальної інформації в блокчейні.

Методи розроблення: спостереження, порівняння, аналіз, синтез; в процесі моделювання використано мову UML; при розробці використано еволюційну модель.

Результати роботи: досліджено поняття блокчейну та обрано блокчейн для реалізації вебзастосунку, проаналізовано програми стейкінгу криптовалют на ринку, досліджено процес розробки смарт-контрактів на даному блокчейні, реалізовано вебзастосунок для стейкінгу криптовалют, що включає в себе реалізацію смарт-контракту та графічного інтерфейсу до нього.

Інструменти розроблення: для побудови смарт-контракту в роботі була використана мова програмування Solidity, для тестування смарт-контракту було використано бібліотеку Chai для Node js, для взаємодії з мережею Ethereum було використано фреймворк Ethers.js, для розгортки мережі Ethereum був використаний інструмент Hardhat, інтерфейс програми реалізовано з використанням фреймворку React.js.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ABI – Application Binary Interface, визначення методів та подій.

API – Application Programming Interface, прикладний програмний інтерфейс;

BEP-20 – Binance Smart Chain Equivalent of ERC-20, стандарт токенів на основі блокчейну Binance Smart Chain;

BSC – Binance Smart Chain, блокчейн-платформа, розроблена командою Binance;

CSS – Cascading Style Sheets, мова таблиць стилів для оформлення вебсторінок;

DApp – Decentralized Application, децентралізований застосунок;

DeFi – Decentralized Finance, децентралізовані фінансові протоколи;

ECDSA – Elliptic Curve Digital Signature Algorithm;

ETH – Ethereum, платформа на базі блокчейну;

HTML – HyperText Markup Language, мова розмітки гіпертексту для вебсторінок;

ICO – Initial Coin Offering, перший публічний продаж токенів для залучення капіталу;

IDO – Initial DEX Offering, ініціальна децентралізована пропозиція токенів;

IDE – Integrated Design Environment, інтегроване середовище розробки;

PoS – Proof of Stake, доказ долі володіння, метод захисту в криптовалюті;

PoW – Proof of Work, алгоритм, який використовується у майнінгу для підтвердження транзакцій шляхом вирішення обчислювальних завдань.

ВСТУП

Оцінка сучасного стану об’єкта розробки. З моменту, коли у 2008 році невідомою особою чи групою людей під іменем Сатоші Накамото (Satoshi Nakamoto) була винайдена перша у світі криптовалюта – Bitcoin, пройшло вже немало часу, проте зацікавленість суспільства у даній темі тільки росте з кожним роком [1].

На сьогоднішній день капіталізація криптовалюти Bitcoin уже в декілька разів перевищує капіталізацію давно наявних на ринку компаній, таких, як Adobe, Siemens, Cisco, Nike, Pfizer, McDonald та інших. Через такі показники на криптовалютному ринку з великою швидкістю з’являються нові рішення, ідеї та тенденції. Питанням дослідження технологій блокчейну та питанням розвитку та функціонування криптовалют присвячені праці багатьох авторів. Наприклад, в роботах С. Накамото [1] введено саме поняття криптовалюта та блокчейн, криптографічні методи шифрування та цифрових підписів розглядаються в працях таких авторів, як О.Маринич [2], Р. Чж, Р. Сюе, Л. Лю [3], дослідженням алгоритмів конченсусу присвячені праці М. Якобсона, Дж. Арі, І. Бентова, А. Габізона, А. Мізрахі та інших [4-7].

Актуальність роботи та підстави для її виконання. Стейкінг є одним зі способів отримання доходу на ринку криптовалюти. Його суть полягає у блокуванні частини своєї криптовалюти на визначений час, щоб підтримувати мережу та отримувати відсоткову винагороду за це. Наразі на ринку існують готові реалізації для стейкінгу криптовалютних активів. Кожен із цих застосунків має свої переваги та недоліки, проте переважна більшість з них підв’язана або ж до криптовалютних бірж, або ж до DeFi. Тривале зберігання власних коштів на криптовалютних біржах не є безпечним через те, що вони можуть мати проблеми із забезпеченням своєї фінансової стійкості, що своєю чергою може призвести до банкрутства біржі та втрати коштів користувачів.

В останні декілька років криптовалютні активи набули такої великої популярності, що почали визнаватися навіть на державному рівні у багатьох країнах [8].

15 березня 2022 року Президентом України В.А. Зеленським підписано Закон України «Про віртуальні активи» [9]. Цим законом в Україні врегульовуються правовідносини, що пов’язані з оборотом віртуальних активів, в тому числі та такому їх підвиду, як криптовалюта.

Відповідно до п.1 розділу VI Закону України «Про віртуальні активи»: «Цей Закон набирає чинності з дня набрання чинності законом України про внесення змін до Податкового кодексу України щодо особливостей оподаткування операцій з віртуальними активами, але не раніше дня опублікування цього Закону» [9]. Цим законом виводиться з тіні криптовалютний ринок, як один із найбільш перспективних напрямів обігу віртуальних активів в Україні. Що робити актуальною задачу розробки прозорих застосунків для роботи в блокчейні.

Таким чином, задача розроблення вебзастосунку для стейкінгу криптовалютних активів є важливою і актуальною.

Мета й завдання роботи. Метою кваліфікаційної роботи є розробка децентралізованого сервісу зі стейкінгу криптовалют, що дасть змогу користувачу отримувати стабільний пасивний дохід за зберігання коштів.

Для виконання поставленої мети в роботі визначено наступні завдання:

1. Дослідити наявні блокчейни для подальшої роботи з ними.
2. Дослідити сучасний стан інформаційних технологій в галузі стейкінгу криптовалют.
3. Розробити смарт-контракт для децентралізованого зберігання інформації про транзакцію користувача.
4. Реалізувати графічний вебінтерфейс для застосунку стейкінгу криптовалют.
5. Проаналізувати результат роботи.

Об’єкт, методи й засоби розробки. Об’єктом дипломної роботи є процеси стейкінгу криптовалют. Предметом роботи є децентралізований вебзастосунок для стейкінгу криптовалюти. Для написання смарт-контракту було використано мову програмування Solidity. Для написання інтерфейсу програми було використано фреймворк React.js. Для створення проєкту було використано середовище програмування Visual Studio Code 2020.

В кваліфікаційній роботі використано такі методи дослідження, як спостереження, порівняння, аналіз та синтез. В процесі розробки вебзастосунку для стейкінгу криптовалют використано еволюційна модель. Для побудови алгоритмів стейкінгу використано хеш-функції, моделювання та мова UML.

Можливі сфери застосування. Основною сферою застосування розробленого продукту є використання для стейкінгу криптовалют з метою отримання пасивного доходу.

РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ БЛОКЧЕЙН

1.1. Технології Блокчейн та Стейкінг

Блокчейн (англ. blockchain, від block – блок, chain – ланцюг, тобто ланцюжок блоків) – це розподілена децентралізована система, яка зберігає постійний запис (ланцюжок) транзакцій, відомостей або даних у вигляді блоків. Кожен блок містить інформацію про транзакції, включаючи попередній блок, унікальний ідентифікатор (хеш) та підтвердження. Ця інформація розподіляється та зберігається на багатьох комп'ютерах (вузлах) у мережі, що робить блокчейн незмінним та стійким до цензури [1].

Одна з ключових особливостей блокчейну полягає в його децентралізації (див. рис. 1), де жоден окремий суб'єкт не контролює весь блокчейн. Замість цього, мережа базується на протоколах, що дозволяють вузлам спільно підтверджувати та підтримувати єдність даних. Це забезпечує безпеку, надійність та прозорість системи, оскільки для зміни будь-якого блоку потрібно бути прийнятим більшістю учасників мережі.

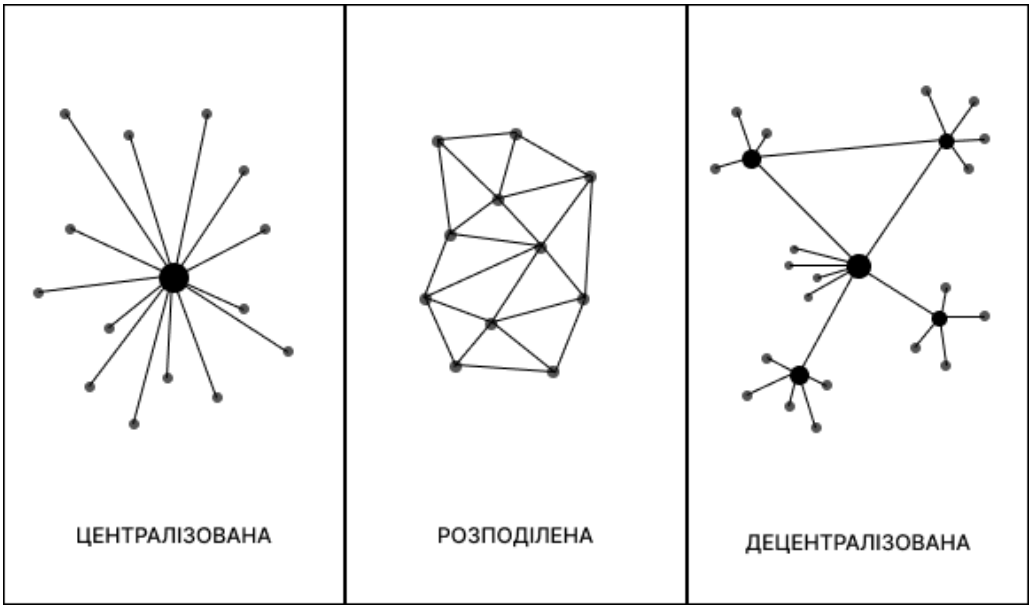


Рисунок 1 – Схеми мережі

Централізована система в криптовалюті означає, що прийняття рішень і управління повністю знаходиться в центральній організації або сервері. У цьому випадку ця сутність або сервер контролює всі аспекти мережі, включаючи транзакції, перевірку, зберігання даних і прийняття рішень.

Розподілена система в криптовалюті означає, що дані та обробка розподіляються між різними вузлами або комп’ютерами в мережі, але належать одному об'єкту та контролюються ним. У цій системі вузли працюють незалежно один від одного, обмінюються даними та співпрацюють для досягнення консенсусу та підтвердження транзакцій.

Децентралізована система в криптовалюті йде ще далі, прагнучи взагалі не мати централізованого пункту контролю та управління. У цій системі рішення приймаються колективно, і кожен вузол мережі має однакові права для участі в процесі прийняття рішень.

Блокчейн широко використовується для реалізації криптовалют, таких як Біткоїн, а також для розвитку різноманітних децентралізованих додатків (DApps), смарт-контрактів та інших інноваційних рішень, що базуються на технології блокчейн.

Технологія блокчейн є однією з основних складових технології криптовалют і широко застосовується для розробки децентралізованих додатків і систем. Блокчейн забезпечує безпеку і надійність даних шляхом застосування криптографічних методів із забезпеченням ідентифікації користувачів і підтвердженням транзакцій.

Блокчейн складається з ланцюжка блоків, кожен з яких містить у собі інформацію про попередній блок та нові транзакції. Кожен блок має свій унікальний хеш, який залежить від його вмісту, та підпис цифрового підпису, який підтверджує автентичність транзакцій (див. рис. 2).

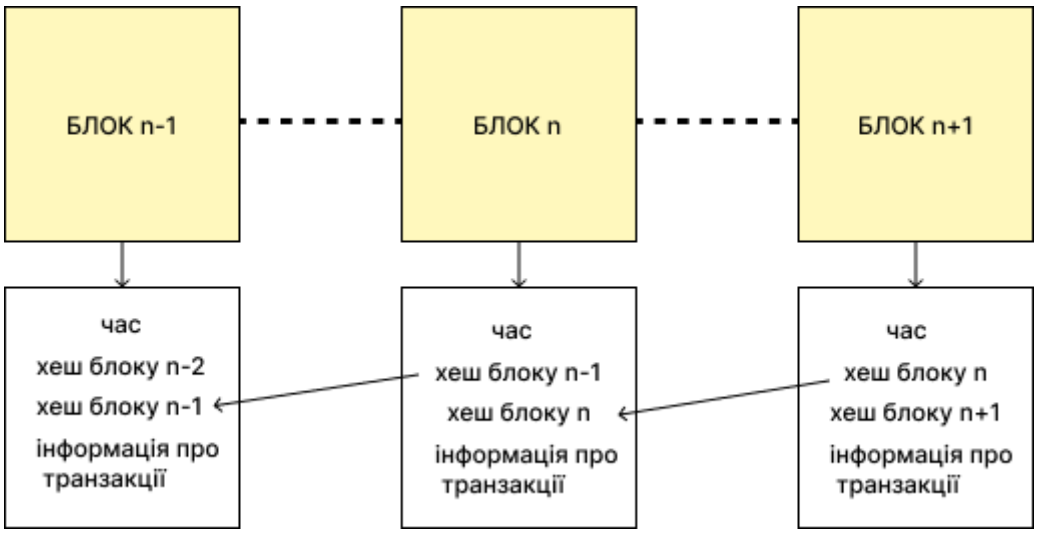


Рисунок 2 – Структура блокчейну

Одним з основних принципів блокчейну є забезпечення відкритості та достовірності даних, навіть у відсутності довіри до інших учасників мережі. Це досягається за допомогою криптографічних методів шифрування та цифрових підписів [2, 3].

Розрахунок криптографічних ключів здійснюється за допомогою односторонніх хеш-функцій [2, 3]. Односторонні хеш-функції є криптографічними функціями, які приймають вхідні дані будь-якої довжини та генерують фіксований вихід, який є унікальним для кожних конкретних вхідних даних. Важливою особливістю односторонніх хеш-функцій є те, що неможливо відновити вхідні дані за вихідним хешем. Це називається "односторонністю" або "незворотністю" хеш-функцій [3].

Неможливо використати інший набір даних, що дає такий самий ключ, оскільки односторонні хеш-функції генерують унікальний хеш для кожного входу.

Втрата криптографічного ключа, як правило, призводить до втрати можливості розшифрування або підпису даних, зашифрованих або підписаних за його допомогою. При цьому, самі дані, що знаходяться в блокчейні, залишаються недоступними без відповідного ключа.

Стейкінг криптовалют – це процес утримання і блокування певної кількості криптовалюти в мережі блокчейн з метою забезпечення безпеки та

підтримки функціонування мережі [4]. Власники криптовалюти, які віддають її у стейкінг, отримують винагороду за свою участь у процесі підтвердження транзакцій та голосування за рішення, пов'язані з управлінням мережею.

У блокчейні, стейкінг зазвичай базується на алгоритмах консенсусу Proof of Stake (PoS).

Стейкінг відіграє важливу роль у забезпеченні безпеки, децентралізації та стійкості мережі блокчейн. Він стимулює власників криптовалют утримувати та активно брати участь у мережі, що забезпечує підтримку та стабільність системи.

1.2. Алгоритми консенсусу PoW та PoS

Proof of Work (PoW) [5] – це механізм запобігання подвійним витратам. Більшість основних криптовалют використовують його як алгоритм консенсусу

Proof-of-Stake (PoS) [6, 7] – метод захисту в криптовалютах.

Алгоритми консенсусу PoW використовуються для підтвердження транзакцій та створення нових блоків в блокчейні шляхом розв'язання складних обчислювальних задач. У системах PoW, майнери, які мають доступ до великої обчислювальної потужності, змагаються між собою за право створювати нові блоки.

Основні етапи алгоритму PoW наступні (див. додаток А):

- Обчислювальна задача: майнери змагаються у розв'язанні складної обчислювальної задачі, яка вимагає значних обчислювальних ресурсів. Ця задача зазвичай базується на пошуку "нульового" хешу шляхом зміни значень неконтрольованих даних, що входять у блок.
- Розв'язання задачі: майнери витрачають обчислювальну потужність, роблячи багато спроб для знаходження правильного рішення. Перший майнер, який знаходить правильний хеш, має право створити новий блок та отримати винагороду.

- Підтвердження блоку: решта мережі перевіряє правильність нового блоку, виконуючи обчислення тієї ж самої задачі. Якщо блок визнається правильним, він додається до блокчейну.
- Винагорода: майнер, який успішно створив новий блок, отримує винагороду в криптовалюті, яка виплачується за його внесок у підтримку мережі та розв'язання обчислювальної задачі.

Стейкінг криптовалют не використовує алгоритм PoW, оскільки вимагає значно менше обчислювальних ресурсів. Замість цього, стейкінг базується на алгоритмі PoS.

В стейкінгу криптовалют з використанням алгоритму PoS учасники, які володіють та блокують певну кількість криптовалюти, мають право створювати та підтверджувати блоки. У порівнянні з алгоритмом PoW, стейкінг не потребує великої обчислювальної потужності та спеціального обладнання.

В стейкінгу PoS використовуються такі етапи (див. додаток Б):

- Stake (Стейк): учасники блокчейну повинні заблокувати певну кількість криптовалюти як їх стейк. Це дія демонструє їхнє зацікавлення в підтримці мережі та забезпечує вагу їхньої участі в процесі консенсусу.
- Вибір блоку: новий блок обирається на основі алгоритму, який враховує різні фактори, такі як розмір стейка, вік стейка та інші параметри. Учасники з більшим стейком мають більшу ймовірність бути обраними для створення нового блоку.
- Підтвердження блоку: власник стейка, який був обраний, має право створити та підписати новий блок. Інші учасники мережі перевіряють цей блок і його легітимність. Якщо блок визнається правильним, він додається до блокчейну.

- Винагорода: учасники, які створюють та підтверджують блоки, отримують винагороду за свою участь у мережі. Винагорода залежить від розміру їхнього стейка та правил протоколу.

1.3. Концепція Web 3.0

Наприкінці 2000-х років один із творців World Wide Web – Тім Бернерс-Лі – запропонував концепцію наступного етапу еволюції Інтернету, що мала назву Web 3.0. Вона стала логічним продовженням своїх попередників Web 1.0 та Web 2.0.

Першу концепцію та початковий етап розвитку Інтернету називається Web 1.0 [10]. Ця фаза першої ітерації мережі тривала з 1991 по 2004 рік. Web 1.0 в першу чергу визначається, як спосіб публікації інформації в мережі, користувачі якої є пасивними споживачами. Основною концепцією Web 1.0 є Read Only (тільки читання); це означає, що користувачі можуть лише переглядати сторінку та взаємодіяти із вмістом. Також у Web 1.0 не існує авторизації, трекерів і реєстрацій.

Ключові особливості Web 1.0 включають:

- Відсутність соціальних мереж: Web 1.0 не вистачало соціальних мереж, де користувачі могли б взаємодіяти один з одним.
- Централізований контроль: сайти та контент контролюються окремими організаціями, які контролюють доступ до інформації та визначають правила взаємодії.
- Відсутність персоналізації: Web 1.0 не забезпечував ні персоналізованого досвіду, ні можливості адаптувати вміст до індивідуальних потреб користувача.

Наступним етапом у розвитку Інтернету став Web 2.0 [11]. Ця епоха почалася у 2005 році, коли американський видавець Тім О'Райлі опублікував статтю з назвою “What Is Web 2.0”. У матеріалі О'Райлі чітко розмежовує Web 1.0 і Web 2.0, а також окреслює вектори подальшого розвитку Інтернету.

Основна відмінність між Web 2.0 та його попередником, Web 1.0, полягає в тому, що Web 2.0 зосереджується на взаємодії та співпраці з користувачем, а не просто на перегляді статичних сторінок.

Основні особливості Web 2.0:

- Взаємодія з користувачем: Web 2.0 дозволяє користувачам не тільки споживати контент, але й створювати його, взаємодіяти один з одним і впливати на його поширення.
- Інтелектуальні системи: Web 2.0 використовує різні інтелектуальні системи, такі як алгоритми рекомендацій, щоб покращити якість вмісту та забезпечити більш персоналізований досвід користувача.
- Мобільність: Web 2.0 дозволяє користувачам взаємодіяти з вебдодатками через мобільні пристрої, роблячи Інтернет доступним у будь-який час і в будь-якому місці.
- Хмарні технології: Web 2.0 використовує хмарні технології для зберігання, обробки та передачі великих обсягів даних.

Остання концепція Web 3.0 [12] спрямована на революцію в тому, як ми сприймаємо та використовуємо Інтернет.

Ключові принципи Web 3.0 включають:

- Семантична інформація: у Web 3.0 інформація має стати більш інтелектуальною та зрозумілою для комп’ютера. Це означає, що дані повинні мати вбудовану семантику, що дозволяє комп’ютерам точніше розуміти та інтерпретувати інформацію.
- Децентралізація: Web 3.0 прагне створити децентралізовану інфраструктуру, де користувачі мають більший контроль над своїми даними та взаємодіють один з одним безпосередньо без посередництва централізованої організації.
- Розширена функціональність: Web 3.0 має надати користувачам нові можливості та функції, такі як смарт-контракти, децентралізовані програми (DApps), взаємодія в реальному часі, персоналізація тощо.

1.4. Огляд технологій Блокчейн

Технології блокчейн, як ідея виникла ще у другій половині XX століття. Одна із найперших згадок про подібну технологію була описана в 1982 році у дисертації американського криптографа Девід Лі Чаум (David Lee Chaum) під назвою «Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups» [13]. В 1991 році два вчені-дослідники Стюарт Хаббер (Stuart Haber) і В. Скотт Сторнетта (W. Scott Stornetta) запровадили обчислювально-практичне рішення для цифрових документів із міткою часу [14]. Ця робота практично не використовувалася, тому у 2004 році патент на неї був втрачений. Минуло ще декілька років, перш ніж у 2008 році невідомою особою, що використовує псевдонім Сатоші Накамото (Satoshi Nakamoto) було опубліковано документ про криптовалюту Bitcoin, що мав назву «Bitcoin: A Peer-to-Peer Electronic Cash System» [1]. У документі описано систему електронної готівки, яка може забезпечити безоплатні, безпечні й безготівкові транзакції між користувачами, обійшовши необхідність у таких посередниках, як банки або ж інші фінансові установи, що зробило транзакції безпечними та анонімними.

Для виконання програми зі стейкінгу криптовалюти спершу потрібно обрати блокчейн на якому буде відбуватися розробка проєкту. На сьогодні існує багато різноманітних блокчейнів, але на мою думку для розробки краще обирати серед найпопулярніших технологій, оскільки вони відзначаються особливою надійністю, та здебільшого є вже перевіреними часом.

Проаналізуємо та порівняємо наступні популярні блокчейни: Bitcoin, Ethereum, Polkadot та Binance Smart Chain.

1.4.1 Блокчейн Bitcoin

Bitcoin – це перший і найвідоміший блокчейн, що був запущений у 2009 році під керівництвом Сатоші Накамото (Satoshi Nakamoto) [1]. Наразі Bitcoin має найбільшу капіталізацію серед криптовалют у світі, а саме, станом на

16.05.2023 року капіталізація становить близько 520 мільярдів доларів [15]. Основна мета Bitcoin – це створити безпечну, надійну та швидку децентралізовану систему електронних платежів.

Серед властивостей блокчейну Bitcoin виділимо наступні:

- Децентралізація: Bitcoin — це децентралізована система. Це означає, що, вона існує без централізованого контролю та управління. Блокчейн Bitcoin зберігається на тисячах комп'ютерів у різних країнах світу, що робить його більш стійким до атак і збоїв.
- Анонімність: Bitcoin дозволяє користувачам здійснювати транзакції анонімно, тобто імена та адреси користувачів не відображаються в блокчейні.
- Безпека: Bitcoin використовує криптографію для захисту від злочинців і шахраїв. Кожна транзакція в блокчейні підписується цифровим підписом, що робить її безпечною та неможливою для злому.
- Майнінг: Bitcoin використовує механізм підтвердження роботи, який дозволяє майнерам отримувати винагороду за обробку транзакцій і створення нових блоків у блокчейні.

Не дивлячись на всі переваги цієї криптовалюти, є одна головна проблема – це відсутність реалізації смарт-контрактів на блокчейні Bitcoin. Це означає, що ми не можемо використовувати його для реалізації задуманого вебзастосунку.

1.4.2 Блокчейн Ethereum

Ethereum, другий за популярністю блокчейн після Bitcoin, багато в чому відрізняється від нього. Станом на 16.05.2023 р. Ethereum має капіталізацію майже 220 мільярдів доларів капіталізації [15].

Ethereum розроблений у 2015 році В. Бутеріним з метою створення платформи для розробки децентралізованих додатків на основі смарт-контрактів («розумних контрактів»), які дозволяють виконувати автоматичні функції на основі певних умов.

Серед властивостей блокчейну Ethereum виділимо наступні:

- Масштабованість: головною особливістю Ethereum є його здатність писати смарт-контракти. Крім того, Ethereum підтримує створення власних токенів, що дозволяє розробникам створювати власні криптовалюти на основі блокчейну Ethereum. Ця можливість є надзвичайно важливою для розвитку проєктів на цьому блокчейні, адже це допомагає збирати інвестиції за допомогою ICO та IDO.
- Децентралізація: оскільки Ethereum був створений для підтримки децентралізованих програм, він має велику кількість розробників і спільноту, що дозволяє швидко й ефективно розробляти проєкти. Із запуском блокчейну Ethereum стало значно простіше створювати власні проєкти в криптовалюті, а також відкрився новий потенціал для створення децентралізованих фінансових послуг (DeFi) у мережі Ethereum.
- Безпека: однією з проблем безпеки Ethereum можна вважати те, що смарт-контракти, які використовуються в мережі, можуть містити програмні помилки, які дозволяють зловмисникам здійснювати атаки на мережу. Ці атаки можуть призвести до втрати коштів і навіть доступу до облікового запису. Щоб запобігти таким атакам, наразі вживаються різні заходи, такі як: аудит смарт-контрактів, посилення

безпеки мережевих протоколів і регулярне оновлення протоколів для виправлення виявлених помилок.

- Майнінг: що стосується майнінгу, Ethereum раніше використовував алгоритм Proof of Work (PoW) для захисту мережі та перевірки транзакцій. Він складався з майнерів, які виконували складні математичні завдання, щоб створити новий блок і отримати винагороду в Ethereum. Однак PoW потребував багато енергії, що було шкідливим для навколишнього середовища. У результаті нещодавно блокчейн перейшов на механізм Proof of Stake (PoS). У PoS учасники мережі вносять депозити в криптовалюті, щоб вони були підзвітні мережі. Учасники, що мають таку заставу називаються стейкерами. Вони отримують можливість формувати нові блоки та проводити транзакції в мережі.

1.4.3 Блокчейн Polkadot

Блокчейн Polkadot був запущений у травні 2020 року. Дана платформа була створена для побудови децентралізованих додатків (DApp), які можуть взаємодіяти між собою та обмінюватися даними та активами. Станом на 16.05.2023 р. Polkadot має капіталізацію близько 6 мільярдів доларів капіталізації [15].

Серед властивостей блокчейну Polkadot виділимо наступні:

- Масштабованість: основною особливістю Polkadot є його архітектура, що дозволяє підключення до блокчейну різних підсистем. Ця функція забезпечує широкі можливості для розробки додатків та інтероперабельності між блокчейнами.
- Майнінг: блокчейн Polkadot використовує механізм Proof of Stake (PoS). Крім того, Polkadot підтримує паралельну обробку транзакцій, що дозволяє прискорити обробку великої кількості транзакцій одночасно.

- Децентралізація: Polkadot має вбудовану систему голосування та управління, що дозволяє учасникам мережі голосувати за зміни протоколу та приймати рішення щодо подальшого розвитку системи. Це забезпечує демократичну структуру системи та дозволяє спільноті Polkadot визначати напрямок розвитку мережі.

1.4.4 Блокчейн Binance Smart Chain

Блокчейн Binance Smart Chain (BSC) було запущено у 2020 році командою найбільшої у світі криптовалютної біржі Binance з метою забезпечення вищої швидкості та менших витрат на транзакції у порівнянні з блокчейном Ethereum.

Майнінг: блокчейн BSC так само як Ethereum та Polkadot підтримує технологію PoS, яка забезпечує швидку обробку транзакцій та високу безпеку мережі.

Масштабованість: BSC використовує власний протокол токенів BEP-20, який є сумісним з протоколом Ethereum. Це дозволяє розробникам легко створювати та випускати свої власні токени на BSC.

Децентралізація: BSC прагне до децентралізації, як і більшість блокчейнів. Однак, в порівнянні з Ethereum, BSC використовує більш централізований підхід до контролю мережі.

1.5. Вибір блокчейну для програмної реалізації

Проаналізувавши різні блокчейни, ми можемо зробити висновок, що для розробки даного проєкту найкраще підходить блокчейн Ethereum або Polkadot. Обидва блокчейни підтримують написання смарт-контрактів та створення власних токенів, а також, є повністю децентралізованими. Проте, блокчейн Ethereum існує на ринку значно довше і має більшу капіталізацію [15]. До того ж даний блокчейн займає друге місце по популярності та капіталізації у світі [15], що робить його надійнішим і стабільнішим у порівнянні з

блокчейном Polkadot. Усі ці фактори є дуже важливими компонентами при створенні успішного проєкту. Саме тому з переліку розглянутих блокчейнів для подальшої роботи мною було обрано саме блокчейн Ethereum.

1.6. Аналіз платформ для стейкінгу

1.6.1 Платформа eToro

eToro (див. рис. 3) – це торговельна платформа, що була заснована у 2007 році. Вона надає можливість не лише здійснювати стейкінг криптовалют, а й торгувати традиційними фінансовими інструментами [16]. На платформі eToro доступні різні криптовалюти, за які можна здійснювати стейкінг.

Рисунок 3 – Логотип платформи eToro

Основними перевагами стейкінгу на eToro є:

- Простота використання.
- Можливість отримувати винагороду за участь у мережі блокчейн без необхідності володіння власними криптовалютними гаманцями.

Для початку стейкінгу на eToro потрібно мати акаунт на платформі та наявність певної кількості криптовалюти, на яку потрібно здійснити стейкінг. Після цього можна обрати відповідну криптовалюту з доступних опцій і перейти до сторінки стейкінгу (див. рис. 4).

Одним з недоліків платформи eToro є її комісійна модель, яка включає збереження відсотка від прибутку користувачів у вигляді комісій. Основні аспекти цього недоліку включають наступне:

- Висока комісія: eToro застосовує високі комісії до торгівлі та інших фінансових операцій. Це може призвести до значного зменшення потенційного прибутку користувачів. Комісії можуть бути застосовані як до стейкінгу криптовалют, так і до інших фінансових інструментів на платформі.
- Централізованість платформи eToro: користувачі, які використовуючи дану платформу, повністю віддають контроль над своїми криптовалютними активами третій стороні. Це означає, що користувачі покладаються на платформу для зберігання своїх активів та отримання винагороди за стейкінг.
- Не відображає показники блокчейну: на платформі eToro немає відображення показників Gas, ціни токена ETH та номеру попереднього блоку, що може суттєво вплинути на ціну комісії, яку сплатить користувач.

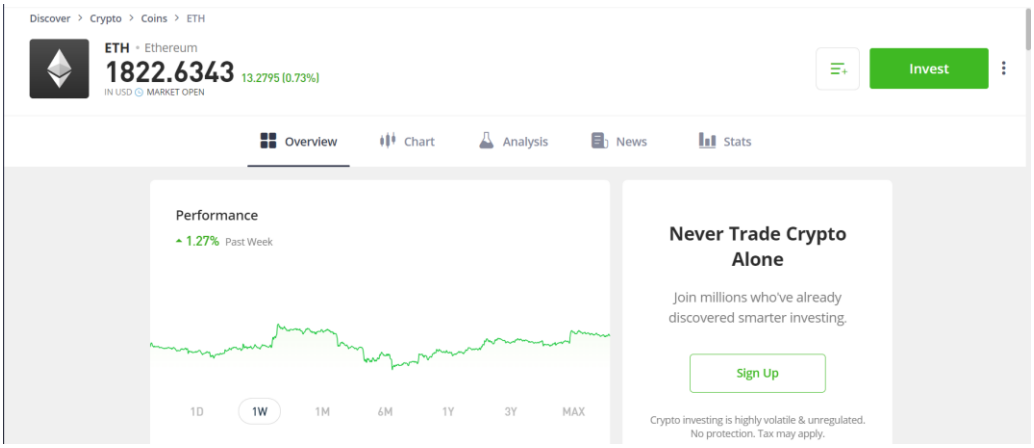


Рисунок 4 – Сторінка стейкінгу платформи eToro

1.6.2 Платформа Coinbase

Coinbase (див. рис. 5) — це криптовалютна платформа, яка була заснована у 2012 році [17]. Вона дозволяє людям купувати, продавати та зберігати різні криптовалютні активи.

Для того, щоб почати взаємодіяти з платформою (див. рис. 6) користувачу спершу потрібно підтвердити свою особу. Після цього користувач може депонувати активи на Coinbase, починаючи з 1 долара та заробляти на стейкінгу криптовалюти до 5% річних.

Рисунок 5 – Логотип платформи Coinbase

Основними перевагами стейкінгу на платформі Coinbase є:

- Можливість скористатися технічною підтримкою
- Наявність вебзастосунку та мобільного додатку.

Водночас платформа Coinbase має декілька недоліків, які варто враховувати:

- Комісії: платформа Coinbase застосовує комісії до операцій купівлі та продажу криптовалют. Це може включати фіксовану комісію або відсоток від суми транзакції. Хоча це стандартна практика для багатьох криптовалютних бірж, комісії можуть збільшувати загальні витрати, особливо при активному трейдингу.
- Мінімальний депозит: платформа Coinbase вимагає мінімальний депозит у розмірі 1 долара США для початку використання платформи. Хоча це дуже низький поріг входу, для деяких користувачів може бути обмеженням, особливо якщо вони хочуть торгувати з великими сумами.

- Централізованість платформи Coinbase: користувачі мають довіряти компанії зберігання своїх криптовалютних активів і особистої інформації. Це може створювати певний ризик з точки зору безпеки, оскільки централізовані системи можуть бути піддані хакерським атакам або порушенню конфіденційності даних.
- Не відображає показники блокчейну: платформа Coinbase не надає інформацію про показники блокчейну, такі як Gas, ціна токену ETH та номер попереднього блоку, що може має вплив на комісію, яку користувач повинен сплатити.

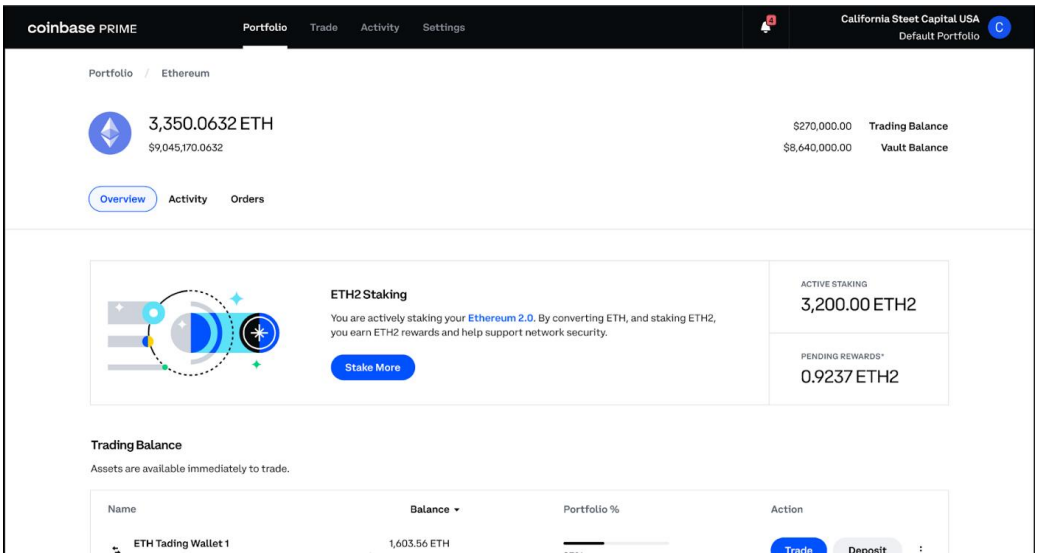


Рисунок 6 – Сторінка стейкінгу платформи Coinbase

1.6.3 Платформа StakeWise

StakeWise (див. рис. 7) – це стейкінг-платформа, яка надає користувачам можливість отримувати винагороду за участь у мережі Ethereu [18]. Платформа StakeWise працює на основі технології DeFi та надає зручні та безпечні рішення для стейкінгу ETH (див. рис. 8).

Рисунок 7 – Логотип платформи StakeWise

Основними перевагами стейкінгу на платформі StakeWise є:

- Моніторинг в реальному часі: платформа StakeWise надає вичерпну інформацію про валідатори, щоб ви були в курсі винагород і ефективності.
- Незалежність від мінімальної суми стейкінгу: платформа StakeWise дозволяє користувачам ставити довільну суму ETH для стейкінгу, навіть якщо вона менша, ніж мінімальна сума, необхідна для самостійного стейкінгу в мережі Ethereum.

Недоліки платформи StakeWise:

- Перенасичений інтерфейс: інтерфейс платформи StakeWise є складним або заповненим багатьма додатковими функціями, це може ускладнити навігацію та використання платформи. Користувачі можуть витрачати багато часу на вивчення інтерфейсу та його функціональності, що може призвести до затримок або незручностей.
- Відсутність гнучкості в управлінні активами: платформа StakeWise пропонує автоматизований процес стейкінгу, що означає, що користувачі не мають повного контролю над своїми активами. Вони не можуть відкликати свої кошти після виконання стейкінгу за межі платформи за власним бажанням.
- Не відображає показники блокчейну: у застосунку платформи StakeWise відсутнє відображення показників блокчейну, таких як рівень Gas, ціна токена ETH та номер попереднього блоку. Дані показники впливають на визначення комісії, яку користувач буде платити.

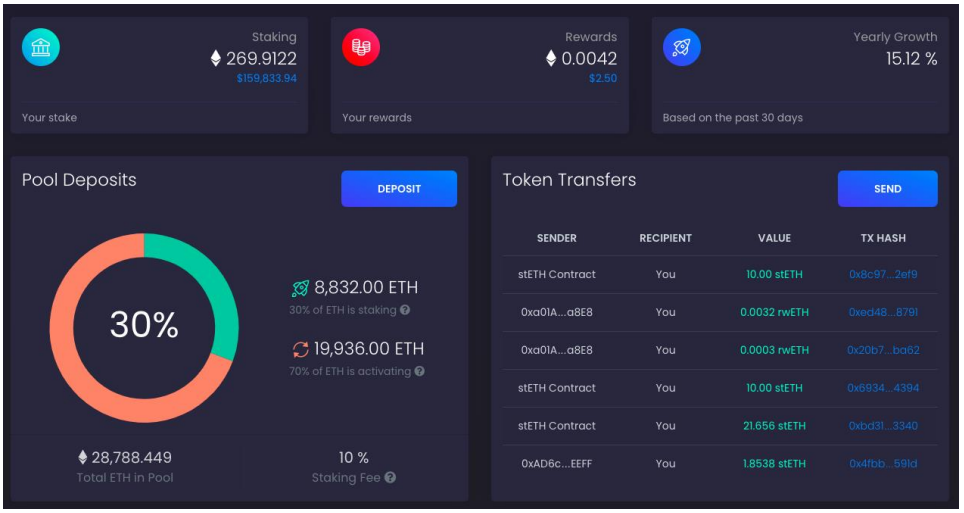


Рисунок 8 – Сторінка стейкінгу платформи StakeWise

Для розробки власного вебзастосунку для стейкінгу криптовалют з урахуванням здійсненого аналізу враховуємо наступні вимоги:

- Децентралізованість: застосунок повинен бути побудований на базі технології блокчейн і мати децентралізовану архітектуру. Це забезпечить безпеку, прозорість та відсутність центральних точок вразливості.
- Зручний інтерфейс: розроблюваний вебзастосунок повинен мати зручний та інтуїтивно зрозумілий інтерфейс, що дозволить користувачам з легкістю використовувати всі функціональні можливості стейкінгу.
- Відсутність мінімального депозиту: застосунок повинен дозволяти користувачам розміщувати на стейкінг довільну суму криптовалюти без обмежень мінімального депозиту.
- Безпека активів: застосунок повинен забезпечувати високий рівень безпеки активів користувачів.
- Відображення важливих показників: розроблюваний застосунок повинен надавати користувачу відображення показників блокчейну, таких як: рівень Gas, ціна токена ETH та номер попереднього блоку в реальному часі.

РОЗДІЛ 2. РОЗРОБКА СМАРТ-КОНТРАКТУ ДЛЯ ВЕБЗАСТОСУНКУ СТЕЙКІНГУ КРИПТОВАЛЮТ

2.1. Огляд технологій розробки

При реалізації вебзастосунку для стейкінгу криптовалют було використано наступні технології:

- мова написання смарт-контракту Solidity [19];
- бібліотека Chai [20] для Node js;
- фреймворк Ethers.js [21];
- інструмент Hardhat [22].

2.1.1 Мова Solidity

При реалізації смарт-контракту для стейкінгу криптовалют було використано мову Solidity. Solidity – це мова програмування високого рівня, заснована на синтаксисі мови JavaScript, яка має синтаксис, подібний до C++. Його головна мета – дозволити розробникам створювати смарт-контракти для платформи Ethereum, які можуть виконувати угоди, зберігати дані та керувати цифровими активами [19].

Особливості мови Solidity [19]:

- Введення: Solidity є статично типізованою мовою, що означає, що кожна змінна має тип, визначений під час оголошення. Підтримуються кілька типів даних, включаючи цілі числа, рядки, масиви, структури тощо.
- Об’єктно-орієнтовані можливості: Solidity підтримує такі концепції об’єктно-орієнтованого програмування, як успадкування, поліморфізм та інкапсуляція.
- Смартконтракт: смартконтракт – це програмний код, який містить умови та дії угоди, яка автоматично виконується. Solidity дозволяє

розробникам створювати смартконтракти, які забезпечують безпечні та надійні транзакції в блокчейні.

2.1.2 Бібліотека Chai

Chai – це бібліотека для тестування коду JavaScript, яка забезпечує читабельний і зрозумілий синтаксис для написання тестів. Він підтримує різні стилі тестування, дозволяючи розробникам вибрати найбільш зручний метод для своїх проєктів. Chai можна використовувати в середовищі та браузері Node.js [20].

Особливості бібліотеки Chai [20]:

- Assertions (ствердження): Chai надає широкий набір стандартних тверджень, які допомагають перевірити поведінку коду. Використовуючи різні методи, такі як assert, expect або should, розробники можуть перевіряти різні аспекти, такі як рівність значень, типи даних, наявність властивостей тощо.
- Розширена функціональність: Chai також дозволяє розширити свою функціональність за допомогою плагінів.
- Підтримує різні стилі тестування: Chai дозволяє розробникам вибрати стиль тестування, який найкраще відповідає їхнім потребам
- Інтеграція з фреймворками: Chai інтегрується з такими популярними фреймворками тестування, як Mocha, Jest, Karma тощо.

2.1.3 Фреймворк Ethers.js

Ethers.js – це фреймворк для взаємодії з блокчейном Ethereum в середовищі JavaScript [21]. Він надає потужні інструменти для створення децентралізованих програм, використання смарт-контрактів і взаємодії з мережею Ethereum.

Особливості фреймворку Ethers.js [21]:

- Взаємодія зі смарт-контрактами: Ethers.js забезпечує зручний спосіб взаємодії зі смарт-контрактами в блокчейні Ethereum. Розробники можуть створювати нові контракти, викликати їхні функції, отримувати дані з контрактів і надсилати транзакції, щоб змінити стан контрактів.
- Робота з гаманцями: Ethers.js дозволяє створювати та керувати гаманцями для зберігання приватних ключів і підписання транзакцій. Це дозволяє розробникам безпечно керувати своїми активами в блокчейні Ethereum.
- Використовуйте смарт-контракти ERC-20: Ethers.js має вбудовану підтримку стандарту ERC-20, що полегшує взаємодію з токенами на основі Ethereum. Розробники можуть отримувати інформацію про баланси, надсилати токени та виконувати інші операції з токенами ERC-20.

2.1.4. Інструмент Hardhat

Hardhat – це інструмент розробки для побудови, тестування та розгортання децентралізованих додатків (DApps) та смарт-контрактів на блокчейні Ethereum [22]. Інструмент надає розробникам зручне середовище для ефективної та безпечної розробки додатків і підтримує широкий спектр функцій для полегшення процесу розробки.

Особливості Hardhat [22]:

- Конфігурація та керування проєктами: Hardhat дозволяє легко створювати та налаштовувати нові проєкти. Розробники можуть визначати параметри збірки, вибирати платформу (наприклад, Ethereum або Binance Smart Chain), налаштовувати мережу та виконувати інші налаштування проєкту.
- Компіляція смарт-контрактів: Hardhat надає інструменти для компіляції смарт-контрактів, перетворюючи їх із мови Solidity у виконуваний код.

Крім того, Hardhat дозволяє розгортати контракти в тестовій мережі та основній мережі Ethereum.

- Розробка та виконання тестів: Hardhat надає потужні можливості для розробки та виконання тестів смарт-контрактів. Розробники можуть створювати автоматизовані тестові сценарії для перевірки правильності виконання контрактів та їх взаємодії з іншими компонентами системи.

2.2. Проєктування застосунку

Призначенням системи є стейкінг криптовалюти користувачем. Діаграму прецедентів наведено в додатку А.

При відвідуванні сторінки користувач може виконати наступні дії:

- "Під'єднати гаманець" (Connect Wallet) – користувач підключає свій гаманець до додатка.
- "Переглянути Gas" (View Gas) – користувач може переглянути поточні витрати на Gas в мережі Ethereum.
- "Переглянути останній блок" (View Last Block) – користувач може переглянути дані останнього блоку в мережі Ethereum.
- "Переглянути ринкові дані" (View Market Data) – користувач може отримати ринкові дані щодо токена ETH.
- "Переглянути винагороди зі стейкінгу" (View Staking Rewards) – користувач може переглянути інформацію про винагороди, які він отримає з різних планів стейкінгу.

Після підключення гаманця, користувач має доступ до наступних дій:

- "Переглянути активи в стейкінгу" (View Staked Assets) – користувач може переглянути інформацію про свої активи в стейкінгу.
- "Підписати транзакцію на зняття" (Sign Withdraw Transaction) – користувач підписує транзакцію на зняття криптовалюти зі стейкінгу.

- "Обрати план стейкінгу" (Select Staking Plan) - користувач обирає план для стейкінгу криптовалюти.

Після підписання транзакції на зняття користувач може:

- "Зняти криптовалюту зі стейкінгу" (Withdraw Staked Cryptocurrency) – користувач знімає криптовалюту зі стейкінгу зі свого гаманця.

Після обрання плану стейкінгу, користувач може виконати наступні дії:

- "Підписати транзакцію на стейкінг" (Sign Staking Transaction) – користувач підписує транзакцію на стейкінг криптовалюти за обраним планом.

Після цього відкривається доступ до:

- "Виконати стейкінг криптовалюти" (Stake Cryptocurrency) – користувач виконує стейкінг криптовалюти за обраним планом.

2.3. Розробка смартконтракту

Розробка смартконтрактів є ключовим аспектом блокчейн технологій, зокрема платформи Ethereum.

Смарт-контракти – це програми, які дозволяють сторонам укладати та виконувати угоди без посередництва та надійно зберігати дані на блокчейні [19]. Вони відкривають безліч можливостей для реалізації різноманітних застосувань, включаючи фінансові послуги, управління активами, логістику, голосування та багато інших сфер. Розробка смарт-контрактів вимагає глибокого розуміння мови програмування, такої як Solidity, та особливостей блокчейн технології.

Для реалізації вебзастосунку зі стейкінгу криптовалюти було обрано мову програмування Solidity [19]. Розроблений смартконтракт мовою Solidity під назвою "Staking" дозволяє користувачам блокувати ефір на визначений період часу, отримуючи при цьому винагороду у визначеній кількості відсотків. Основна ідея контракту полягає в тому, що користувачі можуть здійснювати стейкінг на певний період (30, 90, 180 або 360 днів) з відсотком винагороди (7%, 10%, 12% або 21% відповідно).

Смартконтракт містить структуру "Position", яка зберігає дані про кожен стейкінг, включаючи ідентифікатор, адресу гаманця користувача, дату створення, дату розблокування, процент відсотка, суму стейкінгу, суму відсотків та статус відкритості.

В конструкторі контракту визначені початкові значення відсоткових ставок та періодів блокування, а також адреса власника контракту.

Користувачі можуть викликати функцію "stakeEther" (див. рис. 9) для ставки ефіру на визначений період. Після перевірки, що вибраний період стейкінгу існує, створюється новий стейкінг із вказаними параметрами, а інформація про стейкінг зберігається в кортежі "positions". Також додаються посилання на ідентифікатор стейкінгу для кожної адреси гаманця користувача.

```
function stakeEther(uint numDays) external payable {
    require(tiers[numDays] > 0, "ERROR! No such mapping.");

    positions[currentPositionId] = Position(
        currentPositionId,
        msg.sender,
        block.timestamp,
        block.timestamp + (numDays * 1 days), //* 1 days requires by solidity
        tiers[numDays],
        msg.value, //ether Staked
        calculateInterest(tiers[numDays], numDays, msg.value),
        true
    );

    positionIdsByAddress[msg.sender].push(currentPositionId);
    currentPositionId += 1;
}
```

Рисунок 9 – Функція "stakeEther"

Функція "calculateInterest" (див. рис. 10) обчислює суму відсотків на основі вказаних параметрів відсоткової базисної точки, періоду та суми стейкінгу.

```
function calculateInterest(uint basisPoints, uint numDays, uint weiAmount) private pure returns(uint)
    return basisPoints * weiAmount / 10000;
}
```

Рисунок 10 – Функція "calculateInterest"

Функція "modifyLockPeriods" (див. рис. 11) дозволяє власнику контракту змінювати періоди стейкінгу та відсоткові базисні точки.

```
function modifyLockPeriods(uint numDays, uint basisPoints) external {
    require(owner == msg.sender, "ERROR! You can't change a period of staking because you are not the

    tiers[numDays] = basisPoints;
    lockPeriods.push(numDays);
}
```

Рисунок 11 – Функція "modifyLockPeriods"

Інші функції дозволяють отримувати дані про перелік доступних періодів стейкінгу, їх відсотки для вибраного періоду, інформацію про стейкінг за ідентифікатором, ідентифікатор стейкінгу для певної адреси гаманця, змінювати дату розблокування стейкінгу та закривати стейкінг з виплатою суми стейкінгу та відсотків.

2.4. Тестування смарт-контракту

Тестування смарт-контрактів є критично важливою частиною розробки, оскільки смарт-контракти виконують функції в реальному часі та мають вплив на обмін цінностей та фінансову безпеку. Тестування дозволяє виявити та виправити помилки та вразливості перед розгортанням контракту в реальному середовищі блокчейну. Важливо зауважити, що після розгортання контракту змінювати його більше не можна.

Для розробки проєкту було обрано тестування смарт-контракту на фреймворку для тестування смарт-контрактів написаних мовою Solidity – Chai. Тестування розпочинається з підготовки контракту та створення необхідних змінних та об'єктів.

Блок тестів "deploy" (див. рис. 12) перевіряє правильність ініціалізації контракту та його стан після розгортання. Використовуючи функції Chai, ми перевіряємо, чи правильно встановлено власника контракту, а також чи належним чином налаштовані періоди блокування та відсоток стейкінгу.

```
describe('deploy', function () {
  it('should set owner', async function() {
    expect(await staking.owner()).to.equal(signer1.address)
  })
  it('sets up tiers and lockPeriods', async function () {
    expect(await staking.lockPeriods(0)).to.equal(30)
    expect(await staking.lockPeriods(1)).to.equal(90)
    expect(await staking.lockPeriods(2)).to.equal(180)
    expect(await staking.lockPeriods(3)).to.equal(360)

    expect(await staking.tiers(30)).to.equal(700)
    expect(await staking.tiers(90)).to.equal(1000)
    expect(await staking.tiers(180)).to.equal(1200)
    expect(await staking.tiers(360)).to.equal(2100)
  })
})
```

Рисунок 12 – Блок тестів "deploy"

Блок тестів "stakeEther" перевіряє функцію stakeEther, яка дозволяє користувачам блокувати криптовалюту. Тест переконується, що контракт правильно обробляє передані значення та створює нову позицію з необхідними даними, такими як адреса гаманця, дата створення, дата

розблокування тощо. Також перевіряється, чи правильно зберігаються дані про позицію та чи оновлюється значення currentPositionId.

Блок тестів "modifyLockPeriods" (див. рис. 13) перевіряє функцію modifyLockPeriods, яка дозволяє власнику контракту змінювати періоди блокування та відсоткові ставки. Тест перевіряє, чи можна створювати нові періоди блокування та чи можна змінювати вже наявні. Виконується перевірка, чи правильно оновлюються значення в кортежах tiers та lockPeriods.

```
describe('modifyLockPeriods', function() {
  describe('owner', function () {
    it('should create a new lock period', async function () {
      await staking.connect(signer1).modifyLockPeriods(100, 999);

      expect(await staking.tiers(100)).to.equal(999);
      expect(await staking.lockPeriods(4)).to.equal(100);
    })
    it('should modify an existing lock period', async function () {
      await staking.connect(signer1).modifyLockPeriods(30, 150);

      expect(await staking.tiers(30)).to.equal(150);
    })
  })
  describe('non-owner', function () {
    it('reverts', async function () {
      expect(
        staking.connect(signer2).modifyLockPeriods(100, 999)
      ).to.be.revertedWith(
        'Only owner may modify staking periods'
      )
    })
  })
})
```

Рисунок 13 – Блок тестів "modifyLockPeriods"

Блок тестів "getLockPeriods" (див. рис. 14) перевіряє функцію getLockPeriods, яка повертає список всіх періодів блокування. Тест переконується, що повертаються правильні значення періодів блокування та їх порядок.

```
describe('getLockPeriods', function() {
  it('returns all lock periods', async () => {
    const lockPeriods = await staking.getLockPeriods()

    expect(
      lockPeriods.map(v => Number(v._hex))
    ).to.eql(
      [30,90,180,360]
    )
  })
})
```

Рисунок 14 – Блок тестів "getLockPeriods"

Блок тестів "getInterestRate" (див. рис. 15) перевіряє функцію getInterestRate, яка повертає відсоткову ставку для певного періоду блокування. Тест перевіряє, чи правильно повертається відсоткова ставка для заданого періоду блокування.

```
describe('getInterestRate', function () {
  it('returns the interest rate for a specific lockPeriod', async () => {
    const interestRate = await staking.getInterestRate(30)
    expect(interestRate).to.equal(700)
  })
})
```

Рисунок 15 – Блок тестів "getInterestRate"

Блок тестів "getPositionById" (див. рис. 16) перевіряє функцію getPositionById, яка повертає дані про певну позицію за її ідентифікатором. Тест переконується, що повертаються правильні дані про позицію, такі як адреса гаманця, дати створення та розблокування, відсоткова ставка, сума блокування тощо.

```
describe('getPositionById', function() {
  it('returns data about a specific position, given a positionId', async () => {
    const provider = waffle.provider;

    const transferAmount = ethers.utils.parseEther('5')
    const data = { value: transferAmount }
    const transaction = await staking.connect(signer1).stakeEther(90, data)
    const receipt = transaction.wait()
    const block = await provider.getBlock(receipt.blockNumber)

    const position = await staking.connect(signer1.address).getPositionById(0)

    expect(position.positionId).to.equal(0)
    expect(position.walletAddress).to.equal(signer1.address)
    expect(position.createdDate).to.equal(block.timestamp)
    expect(position.unlockDate).to.equal(block.timestamp + (86400 * 90))
    expect(position.percentInterest).to.equal(1000)
    expect(position.weiStaked).to.equal(transferAmount)
    expect(position.weiInterest).to.equal( ethers.BigNumber.from(transferAmount).mul(1000).div(1000
    expect(position.open).to.equal(true)
  })
})
```

Рисунок 16 – Блок тестів "getPositionById"

Блок тестів "getPositionIdsForAddress" (див. рис. 17) перевіряє функцію getPositionIdsForAddress, яка повертає список ідентифікаторів позицій, створених користувачем з вказаною адресою гаманця. Тест перевіряє, чи повертаються правильні ідентифікатори позицій для заданої адреси гаманця.

```
describe('getPositionIdsForAddress', function () {
  it('returns a list of positionIds created by a specific address', async () => {
    let data;
    let transaction;

    data = { value: ethers.utils.parseEther('5')}
    transaction = await staking.connect(signer1).stakeEther(90, data);

    data = { value: ethers.utils.parseEther('10')}
    transaction = await staking.connect(signer1).stakeEther(90, data);

    const positionIds = await staking.getPositionIdsForAddress(signer1.address)

    expect(
      positionIds.map(p => Number(p))
    ).to.eql(
      [0,1]
    )
  })
})
```

Рисунок 17 – Блок тестів "getPositionIdsForAddress"

Блок тестів "changeUnlockDate" (див. рис. 187) перевіряє функцію changeUnlockDate, яка дозволяє власнику контракту змінювати дату розблокування певної позиції. Тест перевіряє, чи можна успішно змінити дату розблокування для позиції та чи оновлюється відповідне значення в структурі позиції.

```
describe('changeUnlockDate', function() {
  describe('owner', function() {
    it('changes the unlockDate', async () => {
      const data = { value: ethers.utils.parseEther('8') }
      const transaction = await staking.connect(signer2).stakeEther(90, data)
      const positionOld = await staking.getPositionById(0)

      const newUnlockDate = positionOld.unlockDate - (86400 * 500)
      await staking.connect(signer1).changeUnlockDate(0, newUnlockDate)
      const positionNew = await staking.getPositionById(0)

      expect(
        positionNew.unlockDate
      ).to.be.equal(
        positionOld.unlockDate - (86400 * 500)
      )
    })
  })
```

Рисунок 18 – Блок тестів "changeUnlockDate"

Блок тестів "closePosition" перевіряє функцію closePosition, яка дозволяє користувачеві закрити позицію. Тест перевіряє, чи можна успішно закрити позицію та, чи відбувається правильне перерахування суми для виплати в залежності від дати розблокування.

Кожен тест виконується у своєму власному середовищі та перевіряє очікувані результати за допомогою функцій Chai, таких як expect. У разі, якщо очікувані результати не збігаються з фактичними, тест видає помилку та вказує на проблемну частину контракту.

Ці тести забезпечують перевірку правильності роботи кожного фрагмента функціональності смартконтракту "Staking". Вони допомагають виявити можливі помилки, забезпечують стабільність та надійність контракту та підвищують рівень довіри до нього.

РОЗДІЛ 3. РОЗРОБКА ГРАФІЧНОГО ВЕБІНТЕРФЕЙСУ ДЛЯ ЗАСТОСУНКУ СТЕЙКІНГУ КРИПТОВАЛЮТ

3.1. Огляд фреймворку для розробки

Графічний вебінтерфейс даного застосунку було розроблено за допомогою наступних технологій:

- фреймворк React.js [23];
- мова стильового оформлення CSS [24];
- інструмент Figma [25].

3.1.1 Фреймворк React.js

React.js – це фреймворк для розробки інтерфейсів користувача та для створення динамічних і ефективних вебдодатків [23]. Він був розроблений Facebook і використовується для розробки великої кількості вебдодатків у всьому світі. React.js базується на концепції компонентного підходу, що робить його дуже гнучким і простим у використанні.

Особливості фреймворка React.js [23]:

- Синтаксис JSX: React.js використовує JSX (JavaScript XML), який дозволяє писати код у HTML-подібному синтаксисі. Це робить код більш зрозумілим і зручним для розробників, а також допомагає зберегти структуру інтерфейсу користувача.
- Компоненти: React.js розділяє вебінтерфейс на незалежні багаторазові компоненти. Це спрощує розробку та підтримку великих проєктів і дозволяє створювати добре організований модульний код.
- Відстеження змін стану: React.js використовує механізм відстеження змін стану, який дозволяє автоматично оновлювати інтерфейс користувача, коли змінюються дані. Це дозволяє реагувати на дії користувача та зміни даних у реальному часі.

3.1.2 Мова стильового оформлення CSS

CSS (каскадні таблиці стилів) – це мова стилів, яка використовується для визначення зовнішнього вигляду вебсторінок [24]. Це дозволяє розробникам визначати стиль елементів HTML, наприклад їх колір, шрифт, розташування, розмір та інші властивості, що впливають на візуальні аспекти сторінки.

Особливості мови CSS [24]:

- Розділення структури та стилю: CSS дозволяє відокремити структуру вебсторінки (HTML) від її стилю. Це дозволяє розробникам легко змінювати зовнішній вигляд сторінки, не змінюючи саму структуру.
- Селектори та правила: CSS використовує селектори для вибору елементів для застосування стилів. За допомогою правил розробники можуть установлювати такі атрибути стилю, як колір тексту, фон, шрифт, відступи тощо.
- Каскадування: CSS має концепцію каскадування, що означає, що атрибути можуть бути успадковані від батьківських елементів і переписані в дочірніх елементах. Це дозволяє створювати гнучкі та модульні стилі.

3.1.3 Інструмент Figma

Figma – це вебінструмент для розробки інтерфейсу користувача та командної співпраці [25]. Він надає широкий спектр функцій для створення та створення прототипів вебдодатків, мобільних додатків та інших цифрових продуктів.

Особливості інструменту Figma [25]:

- Інтерфейс дизайну: Figma надає інтуїтивно зрозумілий інтерфейс, який дозволяє дизайнерам створювати високоякісні макети та прототипи вебсторінок і програм. Інструменти малювання, векторні об’єкти, текстові редактори та інші інструменти сприяють творчому та детальному дизайну.

- Командна робота: Figma була створена з урахуванням специфіки командної роботи. Користувачі можуть одночасно працювати над проєктом, вносити зміни, коментувати та взаємодіяти в режимі реального часу. Це полегшує співпрацю між дизайнерами, розробниками та клієнтами.
- Компоненти та стилі: Figma дозволяє створювати повторно використовувані компоненти та зберігати стилі. Це забезпечує узгодженість дизайну, швидше вносить зміни та спрощує процес масштабування проєкту.

На рис. 19 наведено створений за допомогою інструменту Figma дизайн головної сторінки, розробленого у кваліфікаційній роботі проєкту.

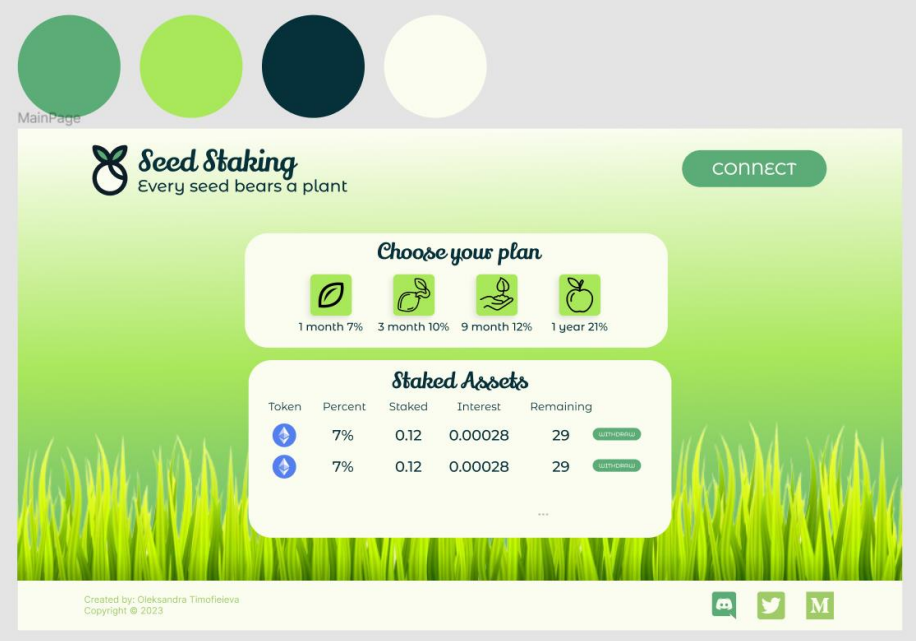


Рисунок 19 – Дизайн проєкту

3.2. Розробка компонентів інтерфейсу

Для розробки графічного інтерфейсу за допомогою фреймворку React.js потрібно поділити сторінку на компоненти з яких вона складається. В цьому випадку маємо компоненти: Header.js, Stake.js, Gas.js, Footer.js, а також основний компонент App.js. Стилізування даних компонентів було виконано за допомогою бібліотеки Bootstrap та розташовано у файлі App.css

3.2.1 Компонента Header

Компонента “Header” – це функціональна компонента React. Основна її функція – це відображення навігаційного рядка на сторінці.

При підключенні компоненти до інших частин додатка можна передати дані (props), такі як функції isConnected() і connect(). Вони виконують наступні ролі:

- isConnected(): ця функція перевіряє, чи встановлене підключення до гаманця у застосунку. Вона використовується для визначення статусу підключення до мережі блокчейн. Залежно від результату, кнопка може відображати різний текст, наприклад, "CONNECTED" або "CONNECT". Таким чином, функція isConnected() допомагає контролювати відображення правильного статусу підключення.
- connect(): ця функція викликається при натисканні на кнопку підключення в компоненті Header. Її основна роль - ініціювати процес підключення до мережі блокчейн. При виклику функція спробує отримати доступ до провайдера (підключення) і підписника (власника підключення). Після успішного встановлення підключення, функція також отримує список активів та іншу необхідну інформацію. Усі ці дії здійснюються з метою підготувати додаток до відображення даних та взаємодії з блокчейном.

На рис.20 наведено фрагмент коду компоненти Header.

```jsx
import React from 'react'

const Header = props => {
  return (
    <>
      <div className="navBar">
        <div className='logoContainer'>
          <img className='logo' src={window.location.origin + '/img/logo.png' }></img>
          <div className='logoSlogan'>
            <span className='logoTxt'>Seed Staking</span>
            <span className='slogan'>Every seed bears a plant!</span>
          </div>
        </div>
        {props.isConnected() ? (
          <div className="connectButton">
            CONNECTED
          </div>
        ) : (
          <div
            onClick={() => props.connect()}
            className="connectButton">
              CONNECT
          </div>
        )}
      </div>
    </>
  )
}

export default Header
```

Рисунок 20 – Фрагмент коду компоненти Header

3.2.2. Компонента Stake

Компонента Stake відповідає за відображення модального вікна, яке дозволяє користувачу виконувати стейкінг ETH. Дана компонента отримує наступні властивості:

- onClose: функція, що викликається при закритті модального вікна. Використовується для повернення компоненті App інформації про закриття модального вікна.
- stakingLength та stakingPercent: ці значення використовуються для відображення терміну та відсотку річного доходу від стейкінгу.
- setAmount: функція, що використовується для збереження введеної користувачем суми.
- stakeEther: дана функція викликається при натисканні кнопки "Stake" і відповідає за виконання операції стейкінгу ETH.

На рис. 21 наведено фрагмент коду компоненти Header

.

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

const Stake = props => {
  const {
    onClose,
    stakingLength,
    stakingPercent,
    setAmount,
    stakeEther,
  } = props

  return (
    <>
      <div className="modal-class" onClick={props.onClose}>
        <div className="modal-content" onClick={e => e.stopPropagation()}>
          <div className="modal-body">
            <h2 className="titleHeader">Stake Ether</h2>

            <div className="row">
              <div className="col-md-9 fieldContainer">
                <input
                  className="inputField"
                  placeholder="0.0"
                  onChange={e => props.setAmount(e.target.value)}
                />
              </div>
              <div className="col-md-3 inputFieldUnitsContainer">
                <span>ETH</span>
              </div>
            </div>

            <div className="row">
              <h6 className="titleHeader stakingTerms">{stakingLength} days @ {stakingPercent} APY</h6>
            </div>
            <div className="row">
              <div
                onClick={() => stakeEther()}
                className="orangeButton"
              >
                Stake
              </div>
            </div>

          </div>
        </div>
      </div>
    </>
  )
}

export default Stake
```

Рисунок 21 – Фрагмент коду компоненти Stake

3.2.3. Компонента Gas

Компонента Gas використовується для отримання та відображення деякої інформації про ціну Gas, останній блок Ethereum та ціну Ethereum в доларах. Ці дані генеруються в наступних функціях:

- getGasPrice: Відправляє POST-запит на URL-адресу API-сервісу Infura з параметрами, щоб отримати поточну ціну Gas. Результат отриманого значення переводиться з Wei у Gwei та зберігається в змінній gasPrice.

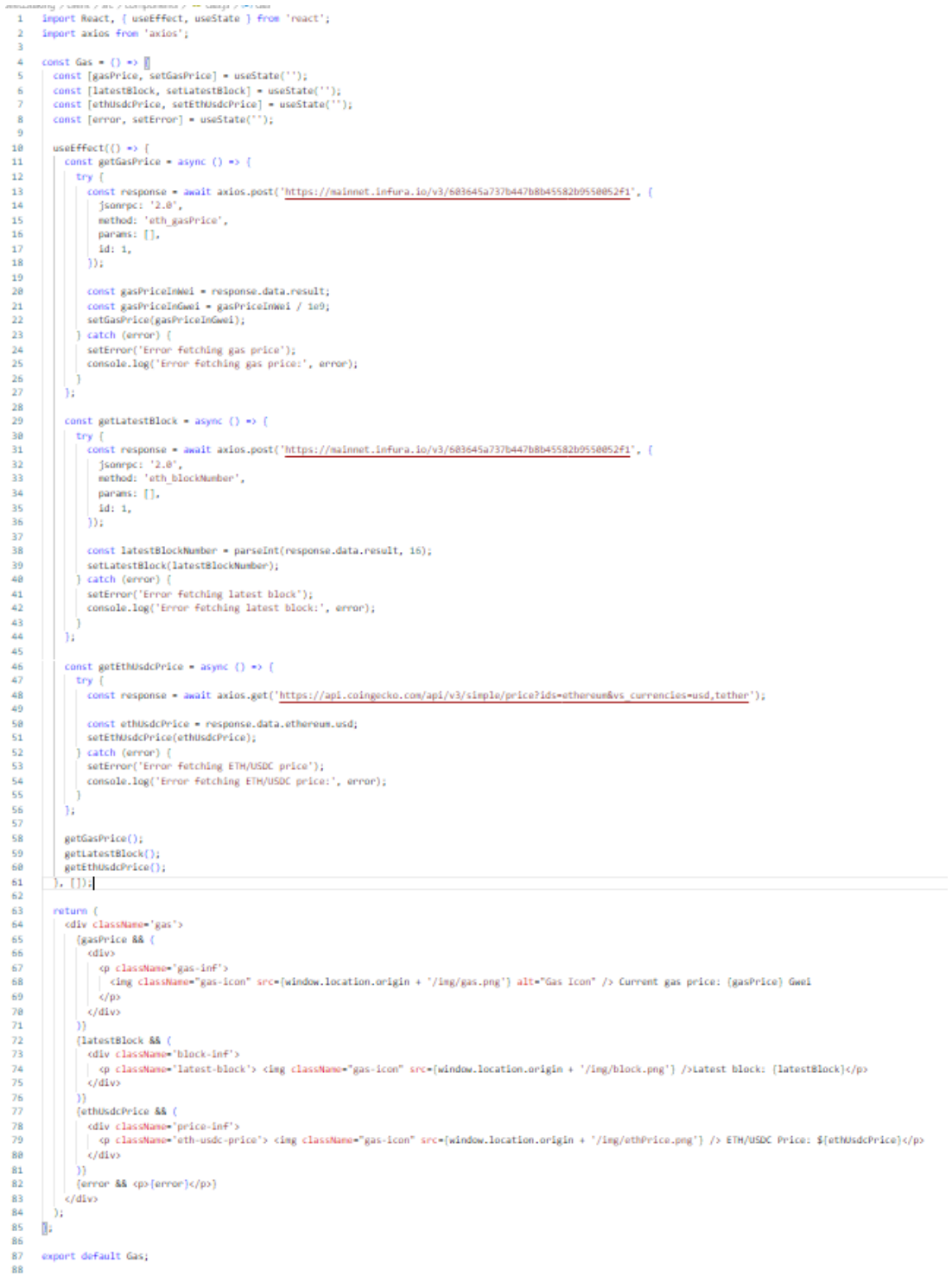
- getLatestBlock: Відправляє POST-запит на URL-адресу API-сервісу Infura з параметрами, щоб отримати номер останнього блоку Ethereum. Отриманий номер переводиться з шістнадцяткового формату у десятковий та зберігається в змінній latestBlock.
- getEthUsdcPrice: Відправляє GET-запит на URL-адресу API-сервісу CoinGecko. для отримання ціни Ethereum в доларах. Отримана ціна зберігається в змінній ethUsdcPrice.

У рендері компонента Gas відображаються різні елементи залежно від наявності отриманих даних та наявності помилок. Якщо gasPrice не є пустим, то відображається поточна ціна Gas. У випадку якщо latestBlock не є пустим, то відображається номер останнього блоку Ethereum. Якщо ethUsdcPrice не є пустим, то відображається ціна ETH в доларах. У випадку помилки в ході виконання коду відображається повідомлення про помилку.

В компоненті Gas використовуються наступні технології:

- Infura – це популярний сервіс, що надає інфраструктуру для взаємодії з блокчейном Ethereum. Infura надає API, яке дозволяє розробникам взаємодіяти з блокчейном Ethereum безпосередньо через їх сервери. Це дозволяє зручно отримувати дані з блокчейну та відправляти транзакції, не потрібно ставити та підтримувати власний вузол Ethereum.
- CoinGecko – це платформа, яка надає інформацію про криптовалюти, наприклад: ціни, ринкові дані та статистику. API-сервіс CoinGecko обробляє запит і повертає відповідь у форматі JSON.

На рис. 22 наведено фрагмент коду компоненти Gas.



```jsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const Gas = () => {
  const [gasPrice, setGasPrice] = useState('');
  const [latestBlock, setLatestBlock] = useState('');
  const [ethUsdcPrice, setEthUsdcPrice] = useState('');
  const [error, setError] = useState('');

  useEffect(() => {
    const getGasPrice = async () => {
      try {
        const response = await axios.post('https://mainnet.infura.io/v3/683645a737b447b8b45582b9550052f1', {
          jsonrpc: '2.0',
          method: 'eth_gasPrice',
          params: [],
          id: 1,
        });

        const gasPriceInWei = response.data.result;
        const gasPriceInGwei = gasPriceInWei / 1e9;
        setGasPrice(gasPriceInGwei);
      } catch (error) {
        setError('Error fetching gas price');
        console.log('Error fetching gas price:', error);
      }
    };

    const getLatestBlock = async () => {
      try {
        const response = await axios.post('https://mainnet.infura.io/v3/683645a737b447b8b45582b9550052f1', {
          jsonrpc: '2.0',
          method: 'eth_blockNumber',
          params: [],
          id: 1,
        });

        const latestBlockNumber = parseInt(response.data.result, 16);
        setLatestBlock(latestBlockNumber);
      } catch (error) {
        setError('Error fetching latest block');
        console.log('Error fetching latest block:', error);
      }
    };

    const getEthUsdcPrice = async () => {
      try {
        const response = await axios.get('https://api.coingecko.com/api/v3/simple/price?ids=ethereum&vs_currencies=usd,tether');

        const ethUsdcPrice = response.data.ethereum.usd;
        setEthUsdcPrice(ethUsdcPrice);
      } catch (error) {
        setError('Error fetching ETH/USDC price');
        console.log('Error fetching ETH/USDC price:', error);
      }
    };

    getGasPrice();
    getLatestBlock();
    getEthUsdcPrice();
  }, []);

  return (
    <div className='gas'>
      {gasPrice && (
        <div>
          <p className='gas-inf'>
            <img className="gas-icon" src={window.location.origin + '/img/gas.png'} alt="Gas Icon" /> Current gas price: {gasPrice} Gwei
          </p>
        </div>
      )}
      {latestBlock && (
        <div className='block-inf'>
          <p className='latest-block'> <img className="gas-icon" src={window.location.origin + '/img/block.png'} />Latest block: {latestBlock}</p>
        </div>
      )}
      {ethUsdcPrice && (
        <div className='price-inf'>
          <p className='eth-usdc-price'> <img className="gas-icon" src={window.location.origin + '/img/ethPrice.png'} /> ETH/USDC Price: ${ethUsdcPrice}</p>
        </div>
      )}
      {error && <p>{error}</p>}
    </div>
  );
};

export default Gas;
```

Рисунок 22 – Фрагмент коду компоненти Gas

3.2.4. Компонента Footer

Компонент Footer дає на вихід елемент з інформацією про розробника проєкту, який потім розміщується в компоненті App.

На рис. 23 наведено фрагмент коду компоненти Footer.

```
1   import React from 'react'
2
3   const Footer = props => {
4     return (
5       <div className='footer'>
6           Created By Oleksandra Timofieieva 2023
7       </div>
8     )
9   }
10
11  export default Footer
```

Рисунок 23 – Код компонента Footer

3.2.5. Компонента App

Компонента App є головною компонентою програми React. В ній відбувається під'єднання до мережі Ethereum за допомогою бібліотеки ethers.js, отримання даних з розумового контракту і відображення їх на вебсторінці.

Основні функції та змінні в компоненті App:

- Створення провайдера (provider): У компоненті App створюється екземпляр ethers.providers.Web3Provider з використанням об'єкта window.ethereum. Цей провайдер дозволяє взаємодіяти з мережею Ethereum через криптовалютний гаманець MetaMask або інші подібні веброзширення.
- Під'єднання до мережі: За допомогою провайдера provider встановлюється з'єднання з мережею Ethereum.
- Отримання підписувача (signer): За допомогою provider.getSigner() отримується підписувач, який може виконувати транзакції в мережі Ethereum від імені користувача.
- Отримання адреси гаманця: Використовуючи signer.getAddress(), отримується адреса гаманця (користувача).

- Взаємодія зі смарт-контрактом: За допомогою ethers.Contract створюється об'єкт контракту, використовуючи адресу контракту та його ABI (визначення методів та подій).
- Виконання операцій з контрактом: За допомогою створеного об'єкта контракту (contract) та підписувача (signer), можна виконувати методи контракту, такі як стейкінг активів (stakeEther()) та вилучення активів (closePosition()).

Компонента App також містить імпорт і рендеринг інших компонентів, таких як Header, Stake, Gas та Footer, які відповідають за різні частини вебсторінки.

РОЗДІЛ 4. ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1. Інструкція користувача

Перед початком роботи з програмою, користувачеві варто переконатися, що він встановив розширення MetaMask для свого браузера та обрав на ньому мережу Ethereum. Криптовалютний гаманець MetaMask дозволяє взаємодіяти з мережею Ethereum та керувати своїми коштами (див. рис 24).

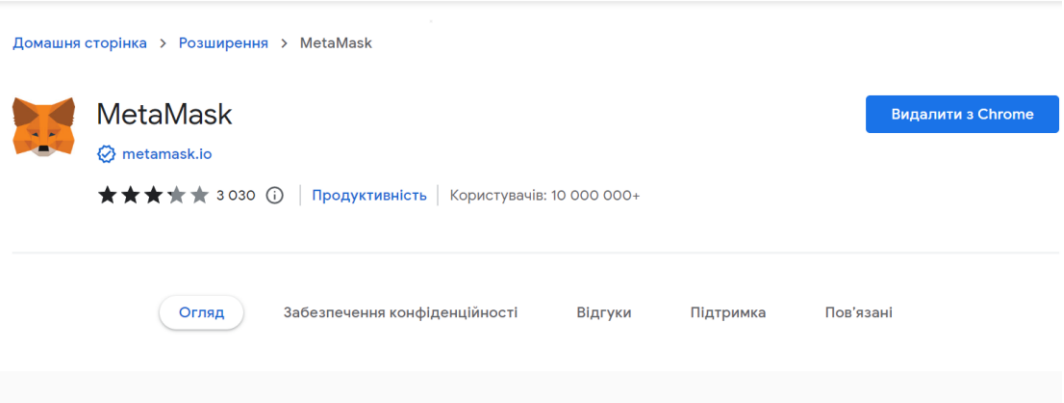


Рисунок 24 – Встановлений гаманець MetaMask

При запуску програми користувач побачить на сторінці різні плани стейкінгу, актуальну інформацію про навантаженість мережі, поточний блок, ціну ETH та кнопку "Connect". Натиснувши на неї, MetaMask запропонує під'єднатися до даного вебзастосунку (див. рис. 25).

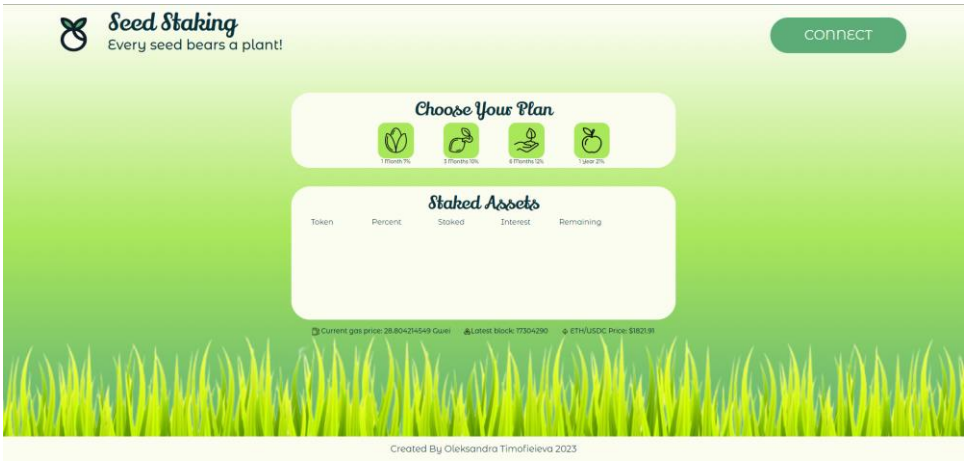


Рисунок 25 – Сторінка до підключення гаманця

Після підключення гаманця, користувачеві відобразяться усі його колишні взаємодії із застосунком у розділі "Staked Assets". В цьому розділі видно розмір стейкінгу, відсоток відсотка, час до кінця стейкінгу, відсоток та опцію вилучення. Якщо стейкінг ще відкритий, ви можете натиснути "Withdraw", щоб вилучити свої активи, а якщо стейкінг вже був закритий, відображатиметься "CLOSED" (див. рис. 26).

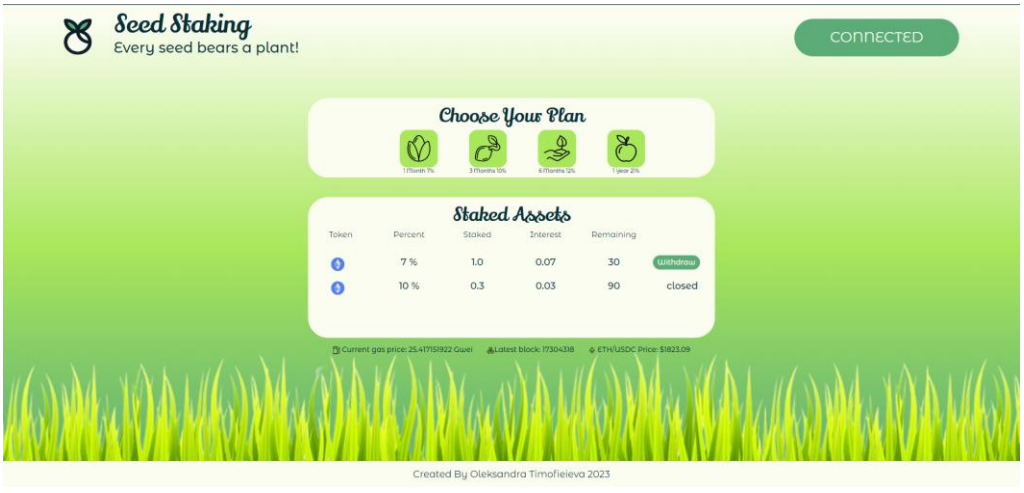


Рисунок 26 – Вебзастосунок після підключення гаманця

Для того, щоб створити новий стейкінг користувачеві потрібно обрати зручний для нього план в розділі “Choose your plan”. Після вибору плану стейкінгу з'явиться модальне вікно, в якому користувачеві потрібно вказати

суму, яку плануєте вкласти (див. рис. 27).

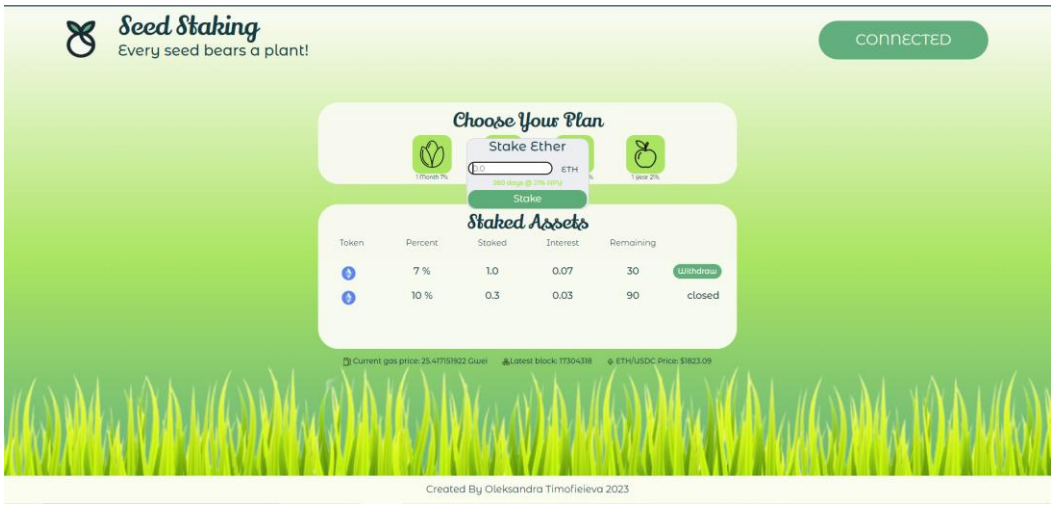


Рисунок 27 – Вебзастосунок під час виконання стейкінгу

Після введення відповідної суми та натискання "Stake" слід буде підтвердити транзакцію в MetaMask (див. рис. 28).

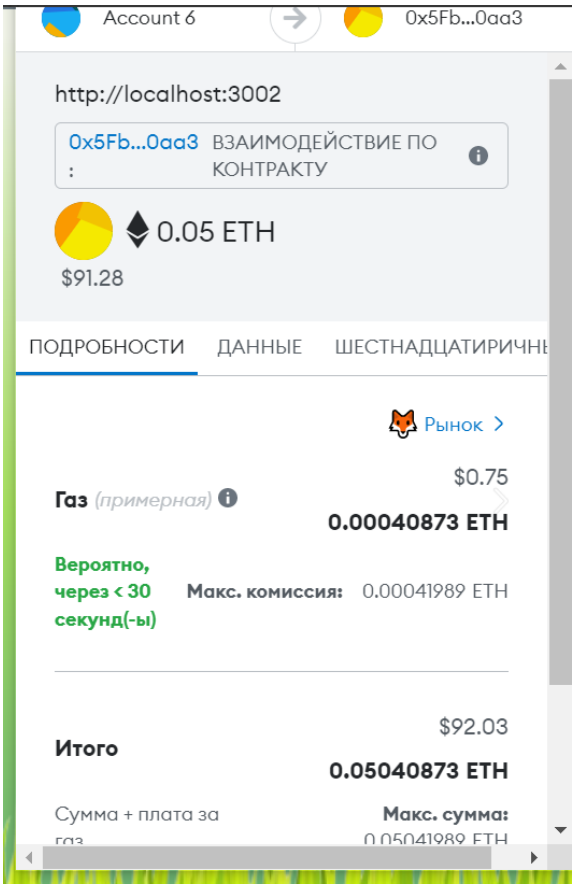


Рисунок 28 – Підписання транзакції на стейкінг

Після успішного підтвердження необхідної транзакції в MetaMask користувач побачить свої активи, у розділі "Staked Assets" (див. рис. 29).

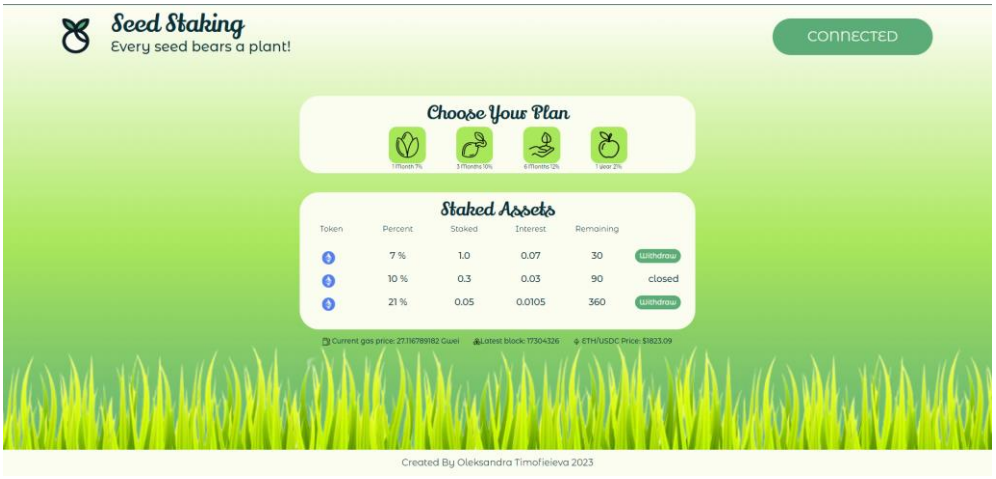


Рисунок 29 – Підписання транзакції на стейкінг

Таким чином, користувач здійснює стейкінг ETH за допомогою вебзастосунку “Seed Staking”.

ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи на здобуття ступеня бакалавра було розроблено децентралізований сервіс зі стейкінгу криптовалют, що дає змогу користувачу отримувати стабільний пасивний дохід за зберігання коштів. З цією метою було здійснено аналіз наявних блокчейнів, досліджено сучасний стан інформаційних технологій в галузі стейкінгу криптовалют, розроблено смарт-контракт для децентралізованого зберігання інформації про транзакцію користувача, реалізовано графічний вебінтерфейс застосунку та протестовано розроблений вебзастосунок.

Проєктування смарт-контракту було ключовим етапом, оскільки він відповідає за реалізацію логіки стейкінгу ETH в мережі Ethereum. В процесі розробки було враховано вимоги безпеки та надійності, а також забезпечено функціональність контракту. Для забезпечення коректності роботи смарт-контракту було здійснено його тестування.

Інтерфейс застосунку розроблено з використанням застосунку Figma, а реалізовано з використанням фреймворку React та мови стилів CSS.

В результаті виконаної роботи можна зробити висновок, що вебзастосунок “Seed Staking” зі стейкінгу ETH в мережі Ethereum є успішно реалізованим проєктом, який забезпечує надійність та зручність для користувачів. Він дозволяє учасникам активно брати участь у мережі Ethereum, забезпечуючи її безпеку та сприяючи розвитку екосистеми. Результатом роботи є функціональний та зрозумілий вебзастосунок, який готовий до використання у реальних умовах.

Перспективними напрямами розвитку проєкту “Seed Staking” є:

- додати більше криптовалютних активів для стейкінгу;
- реалізація стейкінгу додатково в інших мережах;
- реалізація розділу статистики користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 22.05.2023).
2. Маринич О.В. Алгебраїчні структури, криптографія та захист інформації: Електронний навчальний посібник. http://do.unicyb.kiev.ua/marynych/wpcontent/uploads/2020/09/AlgStructCrypto.pdf
3. Zhang R., Xue R., Liu L. Security and privacy on blockchain // ACM Computing Surveys (CSUR). – 2019. – Vol. 52 (3), art. No 51. – P. 30.
4. Що таке стейкінг та як на ньому заробити? [Електронний ресурс] // ForkLog UA. – 804. – Режим доступу до ресурсу: https://forklog.com.ua/exclusive/shho-take-stejking-ta-yak-na-nomu-zarobyty (дата звернення: 22.05.2023).
5. Jakobsson, Markus; Juels, Ari (1999). "Proofs of Work and Bread Pudding Protocols". Secure Information Networks: Communications and Multimedia Security. Kluwer Academic Publishers: 258–272.
6. Як працює алгоритм консенсусу Proof-of-Stake (PoS) і чому він такий популярний? [Електронний ресурс] // MarkupUA. – 2022. – Режим доступу до ресурсу: https://markup-ua.com/yak-pracyuye-algoritm-konsensusu-proof-of-stake-pos-i-chomu-vin-takij-populyarnij/ (дата звернення: 22.05.2023).
7. Bentov, I., Gabizon, A., Mizrahi, A. (2016). Cryptocurrencies Without Proof of Work. In: Clark, J., Meiklejohn, S., Ryan, P., Wallach, D., Brenner, M., Rohloff, K. (eds) Financial Cryptography and Data Security. FC 2016. Lecture Notes in Computer Science(), vol 9604. Springer, Berlin, Heidelberg.
8. Логойда В.М. Перспективи врегулювання правового статусу криптовалюти в Україні. Науковий вісник Ужгородського Національного Університету. Серія Право. 2021. Випуск 63. С. 152-157. URL: https://dspace.uzhnu.edu.ua/jspui/handle/lib/36761
9. Закон України «Про віртуальні активи» №2074-IX від 17.02.2022. URL:

https://zakon.rada.gov.ua/laws/show/2074-20#Text (дата звернення: 22.05.2023).

10. Strickland J. Is There a Web 1.0? [Електронний ресурс] / Jonathan Strickland // Web Design & Development. – 2021. – Режим доступу до ресурсу: https://computer.howstuffworks.com/web-10.htm#pt1 (дата звернення: 22.05.2023).
11. HabibiA., Mukminin A.,PratamaR, Harja H. Predicting Factors Affecting Intention to use Web 2.0 in Learning: Evidence from Science Education. Journal of Baltic Science Education. 2019.Vol. 18, No. 4. P. 595-606
12. Дутчак М. Web 3.0-технології в інтелектуальних освітніх онлайн-платформах / М. Дутчак, І. Лазарович, Ю. Яновський – Ivano-Frankivsk: 19 Scientific Seminar on Innovative Solutions in Software Enginee, 2019. – (19 Scientific Seminar on Innovative Solutions in Software Enginee). – С. 7–9.
13. David Lee Chaum. Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups / David Lee Chaum. – Berkeley: University of California, 1982. – 258 с.
14. Haber, Stuart; Stornetta, W. Scott (January 1991). How to time-stamp a digital document. Journal of Cryptology 3 (2): 99–111. doi:10.1007/bf00196791
15. CryptoSlate [Електронний ресурс] – режим доступу до ресурсу: https://cryptoslate.com/coins/ (дата звернення: 22.05.2023).
16. eToro [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.etoro.com/ (дата звернення: 22.05.2023).
17. Coinbase [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.coinbase.com/ (дата звернення: 22.05.2023).
18. StakeWise [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://stakewise.io/.
19. Chris Dannen. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. — 2017. — С. 256. — ISBN 978-5-8459-1349-4.
20. Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework [Електронний ресурс]

/ ОО – Режим доступу до ресурсу: https://www.chaijs.com/ (дата звернення: 22.05.2023)

21. Joachim Haagen Skeie. Ember.js in Action. — Manning Publications Company, 2014. — 240 p. — ISBN 9781617291456.
22. Hardhat [Електронний ресурс] – Режим доступу до ресурсу: https://hardhat.org/hardhat-runner/docs/getting-started#overview (дата звернення: 22.05.2023).
23. React The library for web and native user interfaces [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/ (дата звернення: 22.05.2023).
24. Learn to style HTML using CSS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/CSS (дата звернення: 22.05.2023).
25. Figma: the collaborative interface design tool [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/ (дата звернення: 22.05.2023).

ДОДАТКИ

Додаток А. Алгоритм PoW

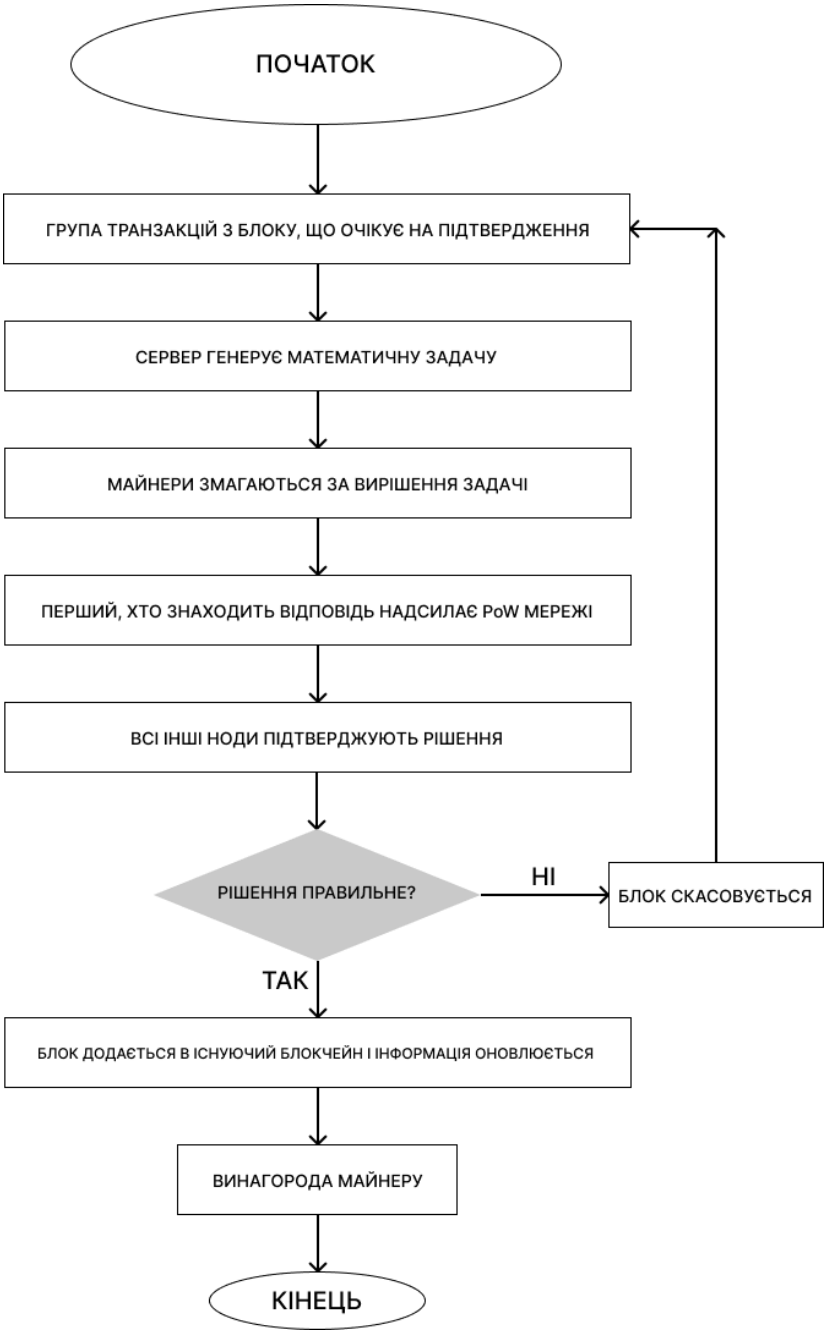


Рисунок А.1 – Алгоритм PoW

Додаток Б. Алгоритм PoS

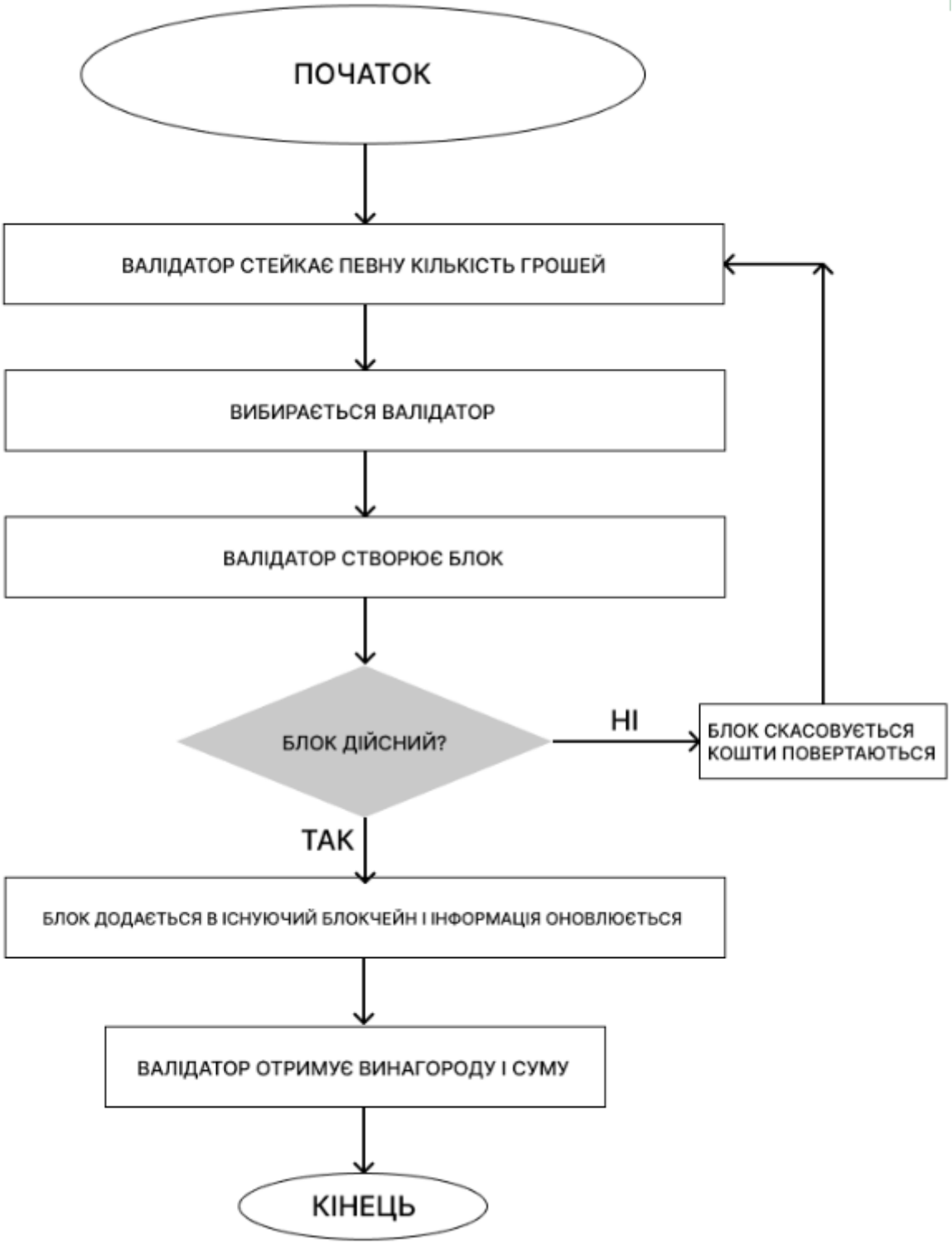


Рисунок Б.1 – Алгоритм PoS

Додаток В. Use-case діаграма проєкту

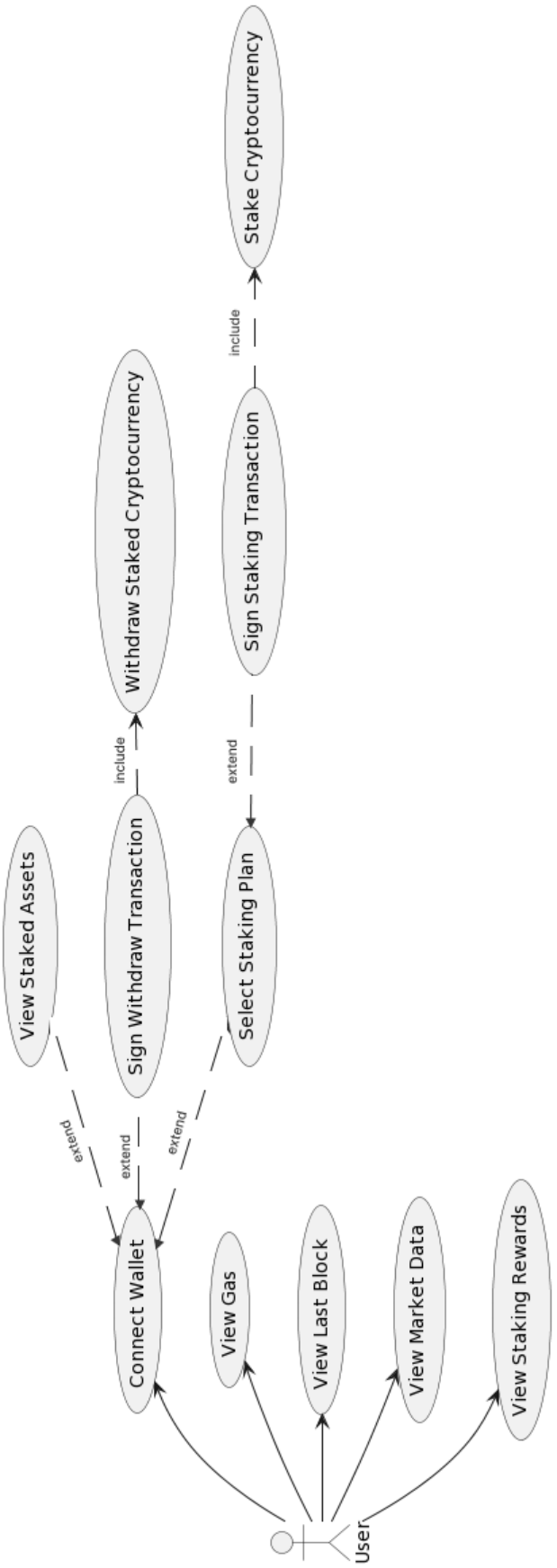


Рисунок В.1 – Use-case діаграма