Зміст

Вступ

DeFi – це екосистема децентралізованих додатків (Dapps), які представляють фінансові послуги, побудовані на базі розподілених мереж без жодного централізованого керівного органу. По суті, DeFi – це набір продуктів і послуг, які заміняють такі фінансові інститути, як банки, страхові компанії, ринки облігацій і грошей [3]. Вона розвивається для створення кращого фінансового ландшафту за допомогою технології Blockchain.

Blockchain – це peer-to-peer технологія, яка зберігає детальну інформацію про всі транзакції, проведені учасниками системи. Вона складається з послідовного ланцюгу блоків, в якому для створення нового ланцюгу необхідно спочатку зчитати всю інформацію про всі попередні старі блоки. Головною особливістю даної технології є можливість розподіленого зберігання інформації [4].

Більша частина створених децентралізованих додатків функціонує на блокчейні Ethereum. Ethereum – це глобальна платформа з відкритим вихідним кодом для Dapps. Основними перевагами є платіжна і клірингова система, доступність, централізація і прозорість [5].

Роботу DeFi-платформи забезпечують криптовалюти. Вони дозволяють здійснювати фінансові операції без посередників, де транзакції проходять набагато швидше з нижчою комісією. Успіх додатків можливий завдяки протоколам, які, в свою чергу, забезпечують всім користувачам рівні можливості у фінансових продуктах, а також дають гарантії щодо прийняття «прозорих» рішень. Протоколи – це смарт-контракти, написані мовою Solidity і завжди відкриті для аудитів.

Смарт-контракти – це програмні контракти, завантажені в децентралізовану мережу Ethereum, які дозволяють двом контрагентам встановлювати умови транзакції без необхідності довіряти її виконання якійсь третій стороні – посереднику. Об’єднані контракти, які виконують надскладні процеси і розрахунки, складають децентралізований додаток [6].

Одиницею цифрового активу створеної платформи є токен – монета, емітована на існуючому блокчейні. Більшість токенів на Ethereum, що є

взаємозамінні, створені за стандартом ERC-20, проте вирізняють ще й токени-жетони для певних додатків, токени-акції, кредитні токени, а також унікальні, які є незамінними.

Беручи до уваги всі переваги децентралізованих систем, а також аналізуючи поточний ринок та бізнес-логіку криптовалютних проєктів, головним завданням дипломного проєкту є створення DeFi-платформи Arcane з відбиттям механізму балансів користувачів на основі ERC-20 токена відповідно до їхніх дій та потреб. Окрім цього, власники мають можливість створювати різнотипні вестінги, включаючи як стандартні, так і опційні. Унікальність проєкту полягає в продуманому бізнес-протоколі, куди входить розроблений токен на основі ERC-20 з розширеними можливостями, який братиме пряму участь у створенні вестінгів – схеми наділення правами на акції.

Для розміщення платформи обрано тестову мережу TESTNET Binance (BscScan). Це мережа для тестування Dapps на BSC. В ній розробники розгортають (деплоять) смарт-контракти з можливістю викликати будь-яку функцію та перевірити коректність її роботи перед офіційним релізом в основній мережі (мейннет), сплативши комісію за допомогою тестової рідної (нативної) валюти BNB. Усі токени, створені на BSC Testnet, не мають реальної цінності, а використовуються лише в цілях тестування готового проєкту [7].

Підключення до мережі відбувається через гаманець в MetaMask, знаючи його адресу та приватний ключ, після чого можна обрати тестову або головну мережу [8].

Розробку додатку Arcane можна поділити на кілька частин: розробка, тестування та деплой контрактів, використовуючи скрипти та різнотипні технології фреймворків Hardhat і Truffle; проведення аудиту: мануальна і тестова частини; розробка документацій; створення front-end частини за допомогою фреймворку Angular тощо.

При розробці проєкту взято до уваги нові технології та останні версії фреймворків та бібліотек OpenZeppelin smart-contracts, Applicature Univarsal Components тощо.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Формування вимог до програмного продукту

Розробити децентралізовану платформу для можливості відображення механізму балансів користувачів на основі ERC-20 токена відповідно до їхніх дій та потреб, здійснення фінансових операцій із залученням токенів для оперування цифровими активами. Застосунок повинен бути зручним та інтуїтивно зрозумілим для користувачів при використанні.

Завдання для повної реалізації, включаючи front-end частину:

– реалізувати процес з’єднання додатку з MetaMask-гаманцем та обрання користувачем потрібної мережі і акаунту для підв’язки;

– автоматичне надання овнеру (деплоєру) всіх прав, включаючи права адміністратора;

– можливість оновлювати дані та додавати нові імплементації контракту, розгортаючи наступні версії за адресою початкового задеплоєного контракту (upgradeable);

– можливість встановлювати нову адресу рутера для отримання точного курсу валют в будь-який момент часу, безпосереднє зв’язування з PancakeSwap контрактами і бібліотеками (незахардкодженість як перевага);

– можливість встановлювати овнеру відсоток комісії залежно від дій користувача, визначаючи напрямок руху токенів та здійснення транзакцій: купівля\продаж токенів або трансфер між транзакціями;

– можливість автоматично обмінювати суму проданих токенів на нативну валюту мережі BNB, встановлюючи також при цьому необхідне значення ліквідності, залежно від значення досягнутого порогу при продажі (поріг встановлюється овнером ще до початку розпродажу);

– можливість додавати певні адреси користувачів овнеру до спеціального списку для отримання нагород при здійсненні купівлі або, навпаки, вилучати їх;

– можливість додавати певні адреси користувачів овнеру до спеціального списку з можливістю здійснювати операції без зняття комісії або, навпаки,

вилучати їх;

– виконувати процес валідації як зі сторони контрактів, так і, при можливості, з боку front-end частини;

– можливість знімати залишки BNB в результаті обміну токенів при досягненні порогового значення;

– реалізувати зняття інших токенів, які користувач міг випадково надіслати на адресу даного контракту;

– реалізувати встановлення можливості обміну токенів та додавання ліквідності при потребі чи бажанні овнера;

– можливість овнером блокувати чи розблоковувати на певний час роботу контракту;

– реалізувати можливість отримувати необхідні дані щодо балансу чи комісій користувачу;

– можливість робити різний розподіл токенів, залежно від того, в якому списку є чи немає адреси отримувача: можливість трансферити без комісій, нараховувати нагороди чи знімати їх;

– можливість взаємодіяти з контрактом безпосередньо відразу на тестовій мережі;

– можливість створювати унікальні вестінги, беручи за основу розроблений токен з розширеними можливостями;

– розробити адаптивний та зручний у використанні інтерфейс користувача для вищезазначених завдань.

Інші завдання:

– завантажити проєкти на сервіс для контролю версіями Git;

– написати NatSpec документацію;

– провести аудит: мануальна та тестова частини;

– створити документацію по використанні проєкту;

– протестувати роботу контрактів за допомогою кількох фреймворків;

– розгорнути смарт-контракти на тестову мережу BscScan.

1.2 Огляд аналогічних додатків

Аналізуючи поточний ринок криптовалютних проєктів, знайдено кілька схожих за бізнес-логікою платформ, зокрема DAO Maker. Їхня логіка полягає у можливості робити стейк для купівлі токенів, так би мовити, депозит в проєкт, а пізніше отримувати пасивний дохід завдяки роботі вестінгу. Інший приклад – Aave-платформа. Вона більш розширена, оскільки за основу має токен, звичайний Staking на його основі, LPStaking і стандартний прямолінійний вестінг. В обох випадках забезпечено можливість розвивати і підтримувати подальші проєкти безпечно, відповідно до протоколу за рахунок продуманого вестінга. Окрім цього, жодна компанія не розгортала контракти на мережі Binance, відповідно в них не є основною нативною криптовалютою BNB.

Тому в результаті аналізу є очевидний факт, що Arcane – унікальна платформа, оскільки користувач має можливість створювати будь-який вестінг на вибір і отримувати лінійний дохід, залежно від часових рамок, за рахунок використання Arcane-токенів, при цьому накопичуючи на балансі певну суму нагород. З часом є можливість потрапити в спеціальний список і здійснювати фінансові операції без додаткових комісій.

Важливим пунктом є те, що в будь-який момент реалізацію контрактів можна розширити до кращих версій і задеплоїти за адресою початкового розгорнутого контракту, що не передбачено в інших аналогічних додатках. Таким чином термін експлуатації та рівень рентабельності є набагато більшим, ніж в інших Dapps.

Реалізований проєкт найкраще підходить благодійним компаніям, які витрачають різні частини коштів на певні потреби і розпорядження йдуть не просто від одного овнера – засновника, а від мультисіга в перспективі – групи співзасновників. Якщо співзасновники вирішать розпочати, для прикладу, певний стартап, то це хороший пункт для безпеки розподілу коштів і гарантованого взяття участі у його подальшій підтримці завдяки бізнес-плану щодо виду і специфіки роботи опційного вестінга.

Перевага додатку над іншими є в тому, що з контрактами можна працювати в самій мережі або створити власний Blockchain, навіть локально, і front-end

частину для відображення всіх дій та балансів спеціально створювати не потрібно.

1.3 Мова програмування розумних контрактів Solidity

Платформа Ethereum представляє фреймворки для високорівневих мов програмування для взаємодії зі смарт-контрактами, такі як Nethereum для .NET-технології, Web3.py – для Python, Web3.js – для JavaScript. Проте для написання смарт-контракту та додавання його в блокчейн-мережу розробники платформи створили спеціальну мову програмування - Solidity.

Solidity – статично типізована, об'єктно-орієнтована та предметно-орієнтована, JavaScript-подібна мова програмування, створена для розробки розумних контрактів, які працюють на віртуальній машині Ethereum (EVM). Програми на мові Solidity транслюються в байткод EVM. Solidity дозволяє розробникам створювати самодостатні програми, що містять бізнес-логіку, результуючу в транзакційні записи блокчейну [9].

На відміну від ECMAScript – стандарту мови програмування, що затверджений міжнародною організацією ECMA згідно зі специфікацією ECMA-262, мова Solidity отримала статичну типізацію змінних і динамічні типи значень. Порівняно з компільованими в такий же байт-код мовами Serpent і Mutan, мова Solidity має важливі відмінності.

Основні переваги мови програмування Solidity:

– підтримуються комплексні змінні контрактів, включаючи довільні ієрархічні відображення (mappings) і структури;

– контракти підтримують спадкування, включаючи множинне і C3-лінеаризацію;

– підтримується бінарний інтерфейс програмування (ABI), що має безліч типобезпечних функцій в кожному контракті;

– специфікована система документування коду, для пояснення послідовності викликів, що отримала назву «Специфікації природною мовою Ethereum» (Ethereum Natural Format Specification).

При розгортанні контрактів слід використовувати останню версію Solidity. Це пов’язано з тим, що регулярно вводяться зміни, а також нові функції та виправлення помилок.

Розробляти програму на Solidity можна на будь-якій операційній системі: Linux, Windows, MacOS X.

Беручи до уваги вищенаведені переваги і особливості мови програмування Solidity, її обрано для написання смарт-контрактів.

1.4 Бібліотека для безпеки розробки контрактів OpenZeppelin

Контракти з бібліотеки OpenZeppelin призначенні для безпеки розробки смарт-контрактів. Вони мають стабільну API, що означає, що створені на їхній основі інші контракти не будуть несподівано порушені під час оновлення до нової другорядної версії.

Установка бібліотеки відбувається за допомогою консольної команди: `npm install @openzeppelin/contracts`. Після установки бібліотеки контракти з неї можна використовувати, імпортуючи їх у свій створений контракт [10].

Щоб забезпечити повну безпеку системи, необхідно використовувати код контрактів, як є, а не копіювати чи вставляти його з якихось онлайн-джерел чи змінювати самостійно.

Важливим пунктом є те, що бібліотека розроблена таким чином, що розгортаються лише ті контракти і функції, які використовує розробник при створенні свого смарт-контракту, тому не відбувається марне витрачання газу на обробку транзакцій.

Якщо потрібно установити бібліотеку upgradeable-контрактів, необхідно ввести в консолі наступну команду: `npm install @openzeppelin/contracts-upgradeable`. Цей пакет відповідає всім нормам створення оновлюваних контрактів: конструктори замінюються функціями ініціалізації, змінні стану ініціалізуються безпосередньо у цій функції.

Розгортання таких контрактів відбувається за допомогою скриптів, написаних на фреймворку Hardhat або Truffle.

Окрім цього, контракти бібліотеки OpenZeppelin надають багато корисних утиліт, які можна безпечно використати у своєму проєкті. Наприклад, з розділу криптографії: перевірка підписів у ланцюжку, перевірка створеного MerkleProof; з розділу інтроінспекції: підтримка інтерфейсів; а також багато утиліт з розділів математики, колекцій, різнотипних трансферів, мультивикликів тощо.

Насправді бібліотека OpenZeppelin має багато напрямів: реалізація таких стандартів, як ERC-20 і ERC-721; гнучка схема дозволів на основі ролей; компоненти для багаторазового використання і створення своїх контрактів тощо.

Враховуючи наведені переваги та можливості, при розробці додатку було обрану дану бібліотеку з можливістю імпорту upgradeable-контрактів.

1.5 Фреймворк для компіляції та тестування Hardhat

Hardhat – це середовище розробки для компіляції, розгортання, тестування та налагодження програмного забезпечення Ethereum. Даний фреймворк допомагає розробникам керувати й автоматизувати повторювані процеси, які стосуються створення смарт-контрактів і Dapps, а також легко впроваджувати більше функціональних можливостей навколо цього робочого процесу: компіляцію, запуск та тестування смарт-контрактів у самому ядрі [11].

Hardhat поставляється зі спеціальною мережею під назвою hardhat. При використанні цієї мережі екземпляр Hardhat Network буде автоматично створено, коли запускається певний таск (завдання), скрипт або при тестуванні смарт-контрактів. Його функціональність зосереджена на налагодженні Solidity, показуючи трасування стека console.log() та явні повідомлення про помилки, коли транзакції невдалі.

Hardhat Runner, команда CLI для взаємодії з Hardhat, є розширюваним запуском завдань. Він розроблений навколо концепцій завдань і плагінів. Завдання можуть викликати інші завдання, що дозволяє визначати складні робочі процеси. Користувачі та плагіни можуть замінювати існуючі завдання, роблячи ці робочі процеси налаштовуваними та розширюваними.

Багато функцій Hardhat надходять від плагінів, тому розробник може самостійно вибирати, які з них буде використовувати. Вбудовані параметри Hardhat за замовчуванням можна перевизначити.

Враховуючи наведені переваги та можливості, при розробці додатку було обрано даний фреймворк для тестування системи при проведенні аудиту.

1.6 Фреймворк для розробки, міграцій та деплою Truffle

Truffle – це середовище для розробки, платформа для тестування та конвеєр активів для блокчейнів з використанням віртуальної машини Ethereum (EVM). Завдяки даному фреймворку можна компілювати, розгортати та здійснювати управління контрактами, також проводити автоматизоване тестування, писати міграційні скрипти, керувати мережею для розгортання великою кількості публічних і приватних мереж, керувати пакетами EthPM і NPM, використовуючи стандарт ERC-190 тощо [12].

Навіть найменший проєкт буде взаємодіяти щонайменше з двома вузлами блокчейну: один на машині розробника, як-от Ganache або Truffle Develop, а інший представлятиме мережу, де розробник в кінцевому підсумку розгорне свою програму (наприклад, головну публічну мережу Ethereum або приватна мережа консорціуму). Truffle надає систему для керування артефактами компіляції та розгортання для кожної мережі, і робить це таким чином, що спрощує остаточне розгортання програми.

Завдяки фреймворку Truffle можна використовувати та розповсюджувати контракти, програмні програми та бібліотеки з підтримкою Ethereum через npm, роблячи свій код доступним для інших, а код інших – собі.

Проєкти, створені за допомогою Truffle, мають певний макет за замовчуванням, що дозволяє використовувати їх як пакети. Цей макет не є обов’язковим, але якщо його використовувати як звичайну конвенцію – або «стандарт де-факто» – то розповсюдження контрактів і програмних програм через NPM стане набагато простішим.

Враховуючи вищенаведені можливості, при розробці платформи було обрано даний фреймворк для побудови смарт-контрактів, написання міграційних скриптів та розгортання на тестовій мережі.

1.7 Тестова Blockchain-мережа TESTNET Binance (BscScan)

Binance Smart Chain – це інноваційне рішення, яке забезпечує програмування та сумісність Binance Chain. Binance Smart Chain покладається на систему з 21 валідатора з консенсусом Proof of Staked Authority (PoSA), який може підтримувати короткий час блокування та нижчі комісії. Найбільш пов’язані валідатори-кандидати стейкінгу є валідаторами та створюють блоки. Виявлення подвійного знака та інша логіка розрізу гарантують безпеку, стабільність і остаточність ланцюга [13].

Binance Smart Chain також підтримує смарт-контракти та протоколи, сумісні з EVM. Передача між ланцюжками та інші способи зв’язку можливі завдяки вбудованій підтримці інтероперабельності. Binance DEX залишається ліквідним місцем обміну активами в обох мережах. Ця подвійна архітектура є ідеальною для користувачів, щоб скористатися перевагами швидкої торгівлі з одного боку та створювати свої децентралізовані програми з іншого.

Основні переваги BSC:

– самостійний блокчейн: забезпечує безпеку з обраними валідаторами;

– сумісний з EVM: підтримує всі наявні інструменти Ethereum разом із швидшою завершеністю та дешевшою платою за транзакції;

– забезпечує ефективний подвійний ланцюг зв'язку;

– оптимізовано для масштабування високопродуктивних Dapps, які вимагають швидкого та безперебійного використання;

– розповсюджується з управлінням на ланцюжку: підтвердження зацікавлених повноважень залучає до децентралізації та учасників спільноти;

– як рідний токен, BNB служить як газом для виконання смарт-контрактів, так і токенами для стейкінгу.

Усі токени, створені на BSC Testnet, не мають реальної цінності, а

використовуються лише в цілях тестування готового проєкту перед релізом на основну мережу.

Враховуючи можливості мережі та наявність схожих додатків, розгорнутих на ній, при розробці Arcane для розгортання було обрано BSC Testnet.

1.8 Мова розмітки HTML

HTML – це стандартизована мова розмітки документів для перегляду веб-сторінок у браузері.

Веб-браузери отримують HTML-документи з веб-сервера або з локальної пам'яті і передають документи в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично, спочатку включаючи сигнали для зовнішнього вигляду документа [14].

Елементи HTML є будівельними блоками сторінок HTML. Мова розмітки сторінок надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Теги, такі як: <input /> <p> оточують і надають інформацію про текст документа і можуть включати інші теги як піделементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML впроваджує засоби для:

– створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;

– отримання інформації із всесвітньої мережі через гіперпосилання;

– створення інтерактивних форм;

– включення зображень, звуку, відео, та інших об'єктів до тексту.

Мову розмітки HTML було вибрано для створення інтерфейсу користувача.

1.9 Скриптова метамова SCSS

SCSS – «діалект» мови SASS. SASS – скриптова метамова, яка

інтерпретується в каскадні таблиці стилів (CSS). Вона призначена для підвищення рівня абстракції коду та спрощення файлів CSS.

Скриптова метамова має два синтаксиси:

– SASS (оригінальний) – відрізняється відсутністю фігурних дужок, в ньому вкладені елементи реалізовані за допомогою відступів, а правила відокремлюються переведенням рядка;

– SCSS (новий) – використовує фігурні дужки (подібно до CSS) [15].

Файли SASS-синтаксису мають розширення .sass, SCSS-синтаксису – .scss.

SASS розширює CSS, надаючи кілька механізмів, доступних в більш традиційних мовах програмування, зокрема об'єктно-орієнтованих мовах, але недоступних для CSS. Інтерпретатор SASS транслює SassScript у блоки правил CSS.

SASS дозволяє визначати змінні. Змінні починаються зі знака долара ($). Присвоєння значень змінних здійснюється за допомогою двокрапки (:). SassScript підтримує чотири типи даних: число, рядок (з лапками чи без), логічний (булевий) тип та колір (ім'я або імена). Змінна може бути аргументом чи результатом однієї чи кількох функцій. Під час трансляції значення змінних вставляються у вихідний (тобто результуючий) документ CSS.

Одна з ключових особливостей SASS – вкладені правила, які полегшують процес створення і редагування вкладених селекторів.

Проаналізувавши всі переваги скриптової метамови SCSS: можливість використання змінних, вкладені правила, можливість використання умовних операторів та циклів, її було вибрано для стилізації інтерфейсу користувача.

1.10 Мова програмування TypeScript

TypeScript – мова програмування, що позиціонується як засіб розробки веб-застосунків і розширює можливості JavaScript.

TypeScript є зворотньо сумісною з JavaScript. Фактично після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js [16].

Основний принцип мови – будь-який код на JavaScript сумісний з TypeScript, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання. Більш того, можна залишити наявні JavaScript-проєкти в незмінному вигляді, а дані про типізації розмістити у вигляді анотацій, які можна помістити в окремі файли, що не заважатимуть розробці і прямому використанню проєкту. Основна перевага TypeScript полягає в тому, що вона може висвітлювати несподівані дії у коді, знижуючи ймовірність помилок.

Окрім цього, у TypeScript два типи сумісні, якщо їхня внутрішня структура сумісна. Це дозволяє реалізувати інтерфейс, просто маючи форму, яку вимагає інтерфейс, без явної імплементації.

Переваги над JavaScript: можливість явного визначення типів (статична типізація); підтримка використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах); підтримка підключення модулів.

Виходячи з цих переваг для розробки front-end частини було вибрано мову програмування TypeScript.

1.11 Мова програмування JavaScript

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [17].

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

– написання сценаріїв веб-сторінок для надання їм інтерактивності;

– створення односторінкових та прогресивних веб-застосунків (AngularJS, Vue.js, React);

– програмування на боці сервера (Node.js(Express.js)) стаціонарних застосунків, мобільних застосунків;

– сценаріїв в прикладних програмах;

– всередині PDF-документів тощо.

1.12 Фреймворк для розробки front-end частини Angular

Angular – написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular – це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників.

Angular представляє фреймворк для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків. В цьому плані Angular є спадкоємцем іншого фреймворка AngularJS. У той же час Angular це не нова версія AngularJS, а принципово новий фреймворк.

До основних переваг фреймворка Angular належать: декларативний стиль коду; використання директив; висока швидкість розробки; SPA (Single page application); модульність; наявність готових рішень; простота тестування [18].

Angular також має такі ES6-можливості, як: анонімні функції; ітератори; цикли типу For/Of; Python-подібні генератори; рефлексія; динамічне завантаження; асинхронна компіляція шаблонів; заміна контролерів та $scope (області видимості) компонентами та директивами – компонент є директивою з шаблоном.

1.13 Бібліотека для front-end розробки Applicature Univarsal Components

Applicature Univarsal Components – це бібліотека, призначена для полегшення розробки Blockchain-проєктів, що є створеною розробниками компанії Applicature. Наразі вона підтримує лише версію Angular 13, проте згодом матиме підтримку

нижчих версій [19].

В основному бібліотека об’єднує компоненти HTML і CSS разом з компонентами Angular, що приносить велику користь як дизайнерам, так і розробникам.

Вона написана на TypeScript і пропонує інтерфейс користувача всього корпоративного класу. Це вважається одним з найоптимальніших рішень у розробці проєкту, а також часто використовується в розробці додатків SASS.

Встановити бібліотеку можна за допомогою консольної команди: `npm install @applicature\styles @applicature\components`.

Наступні налаштування для роботи проводяться у файлах styles.css, polyfills.ts, tsconfig.app.json, package.json, app.module.ts, angular.json, встановлюючи необхідні значення властивостей для подальшої роботи.

2 КОНСТРУКТОРСЬКА ЧАСТИНА

2.1 Аналіз та розроблення бізнес-логіки Arcane-токена проєкту

Arcane-токен – це основний токен розробленого проєкту Arcane, створений на основі ERC-20 стандартного токена. Беручи до уваги за основу розроблення смарт-контракт з бібліотеки OpenZeppelin, визначено, що Arcane-токен є взаємозамінним. Це показує, що в розробленому смарт-контракті є хорошим варіантом створення автоматичного додавання ліквідності до майнінг пула (вклад в загальний фонд проєкту) для забезпечення стабільності виплат користувачам.

Оскільки Arcane-токен повинен включати в себе інтеграцію смарт-контрактів з децентралізованої платформи PancakeSwap для точного визначення курсу валют в будь-який момент часу, додатково він зможе обмінювати рідний токен на основну валюту BNB мережі BscScan Testnet. При обміні за вказаним овнером в період ініціалізації контракту рутером через створену фабрику буде створена пара Arcane:BNB, через яку можна буде здійснювати різноманітні фінансові операції не тільки з токенами проєкту, але й безпосередньо з валютою BNB.

Особливістю смарт-контракту ArcaneToken є upgradeable-властивість. Це означає, що першочергово при розгортанні контракту створюється адреса імплементації (виключно описана логіка смарт-контракту), а також proxy-адреса, що є проксі-сервером, оновити який може вказаний при ініціалізації адміністратор. Після описаних дій proxy-адреса верифікується за адресою імплементації, після чого смарт-контракт готовий до використання. Проте бувають ситуації, коли частково необхідно змінити бізнес-логіку проєкту після його запуску на основну мережу. Тоді смарт-контракт можна змінити, розгорнути нову версію і оновити імплементацію створеної раніше proxy-адреси. В такому випадку найефективнішим буде використання TransparentUpgradeableProxy – тип проксі-сервера [20].

Крім цього, важливим пунктом є передбачення ситуацій, які трапляються через помилки користувача. Для прикладу досить часто є можливий помилковий переказ коштів на баланс контракту: як рідного токена, так і BNB. Для успішного

усунення даної проблеми необхідно створити функцію, яка б дозволяла овнеру знімати ці кошти і надсилати їх назад на призначену адресу. Таке рішення сприяє активним фінансовим циркуляціям і не заблоковує валюти на адресі контракту.

Хорошим рішенням є створення функції для зняття овнером залишків BNB, які залишаються після додавання ліквідності.

Загальна концепція бізнес-логіки проєкту Arcane полягає в тому, що контракт містить певні списки адрес користувачів для нарахування винагород, а також нарахування чи скасування різнотипних комісій. Користувач може купити токен і вкласти його в пул створеного вестінгу або інший криптопроєкт, який співпрацює зі створеним токеном. Фінансові операції здійснюються двома валютами: Arcane-токен та BNB.

Встановлення комісій передбачено для отримання коштів на технологічні потреби та обробку транзакцій. В контракті передбачено 3 види таких комісій: при купівлі токена, при продажі токена та при окремих внутрішніх транзакціях. У випадку, коли користувач активно співпрацює з Arcane-командою, він потрапляє в список користувачів, які можуть здійснювати будь-які можливі операції з контракту без плати комісії. Є можливість також потрапити в список користувачів, яким нараховується певна сума нагороди, залежно від виконаних дій.

Реалізований проєкт найкраще підходить благодійним компаніям, засновники яких працюють в команді. Тоді створюється мультисіг – контракт, який представляє групу осіб. Якщо співзасновники вирішать розпочати, для прикладу, певний стартап, то це хороший пункт для безпеки розподілу коштів і гарантованого взяття участі у його подальшій підтримці завдяки бізнес-плану щодо виду і специфіки роботи опційного вестінга.

2.2 Побудова діаграми вищого рівня та виділення варіантів (сценаріїв) використання

Розглядаючи проєкт з Blockchain-сторони, умовно можна виділити наступні основні частини: смарт-контракт ArcaneToken, Uniswap-інтерфейси, гаманці користувачів, торгові пули (вестінги), а також основна валюта мережі BNB.

Залежно від можливих користувачів та дозволів, встановлених у смарт-контрактах, можна виділити наступні сценарії використання:

– користувач може купити токени (з комісією або без неї, залежно від умов);

– користувач може отримати токени винагороди;

– користувач може зняти з балансу нараховану суму нагород;

– користувач може отримати інформацію, чи включена певна адреса в список додавання чи скасування комісії;

– користувач може отримати інформацію, чи включена певна адреса в список адрес для нарахування нагород;

– користувач може отримати інформацію про час можливого періоду блокування токенів або блокування роботи смарт-контракту;

– користувач може отримати інформацію про встановлену загальну комісію, а також інші види комісій;

– користувач може отримати інформацію про суму, яку він може зняти від бажаної суми з окремим відрахуванням комісії;

– користувач може дізнатись детальніше про поточний власний баланс на рахунку;

– користувач може дізнатись інформацію про встановлені десяткові значення валюти;

– користувач може дізнатись дані щодо загального балансу адреси контракту;

– власник може ініціалізувати контракт після його розгортання на мережі;

– власник може встановлювати порогове значення, при досягненні якого ліквідність буде додаватись автоматично;

– власник може додавати адреси в список нарахування винагород, а також видаляти певні адреси з існуючого списку;

– власник може встановлювати можливі види комісій, а також обмінювати ці значення між собою;

– власник може встановлювати максимальний відсоток комісії при

проведенні транзакцій;

– власник може встановлювати адресу рутера та змінювати її для можливості отримання в певний період точного курсу валют;

– власник може знімати залишки нативної валюти з балансу контракту;

– власник може знімати інші токени з адреси контракту, які могли помилково бути надіслані;

– власник може встановлювати логічне значення для можливості автоматичного додавання ліквідності для контракту;

– власник може встановлювати часові параметри блокування та розблокування токенів, а також завдяки цій дії визначати подальшу роботу контракту.

Гаманці користувачів представляють собою адреси в мережі, на яких зберігаються різноманітні криптовалюти, а також, з яких відбуваються транзакції.

Основна валюта мережі Binance Testnet – BNB Coin буде використана для оплати розгортання смарт-контрактів, а також виконання транзакцій з гаманців користувачів та власника (розробника) Arcane-токена.

High-Level Diagram (діаграму вищого рівня) наведено в графічній частині (додаток А).

2.3 Виділення наслідкових подій у транзакціях

Події у Solidity генеруються смарт-контрактом при виклику певних його методів користувачем або іншим контрактом. Завдяки подіям кожен Blockchain-користувач може отримати коротку інформацію про успішно завершену транзакцію [21].

При розробці Arcane-токена необхідно виділити наступні події при зміні значень важливих змінних:

– `Threshold(uint256 threshold)` – подія, що генерується, коли овнер змінив значення порогу (threshold), при досягненні якого автоматично відбувається обмін між Arcane-токенами та нативною валютою BNB;

– `SwapAndLiquifyEnabledUpdated(bool enabled)` – подія, що

генерується, коли овнер змінює значення, що вказує на можливість здійснення автоматичного обміну між Arcane-токенами та нативною валютою BNB;

– `SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity)` – подія, що генерується, коли здійснився обмін певної суми токенів з балансу користувача на BNB. Може бути використана для відстеження токенів, які залишились на балансі користувача (tokensSwapped), BNB після обміну на балансі контракту (ethReceived), токени, які були використані для обміну (tokensIntoLiquidity);

– `Deliver(address indexed sender, uint256 rAmount, uint256 rTotal, uint256 tFeeTotal)` – подія, що генерується, коли користувач (sender) знімає суму нарахованих для нього нагород (rAmount). При цьому оновлюється значення загальної суми нагород (rTotal) та загальної суми комісій (tFeeTotal);

– `ExcludeFromReward(address indexed account, uint256 tOwned)` – подія, що генерується, коли овнер видаляє адресу користувача (account) зі списку для нарахування нагород і змінює значення поточного балансу користувача, додаючи до попередньої суми суму нарахованих нагород до моменту здійснення даної транзакції (tOwned);

– `IncludeInReward(address indexed account, uint256 tOwned)` – подія, що генерується, коли овнер додає адресу користувача (account) до списку для нарахування нагород і встановлює нульове значення поточного балансу нагород користувача (tOwned);

– `TransferFromSender(address indexed sender, uint256 tOwned, uint256 rOwned)` – подія, що генерується, коли користувач (sender) здійснює пересилання коштів зі своєї адреси на адресу іншого користувача. При цьому змінюється значення його поточного балансу токенів (tOwned), а також змінюється значення балансу нагород (rOwned);

– `TransferToRecipient(address indexed recipient, uint256 tOwned, uint256 rOwned)` – подія, що генерується, коли користувач (recipient) отримує кошти, переслані з адреси іншого користувача. При

цьому змінюється значення його поточного балансу токенів (tOwned), а також змінюється значення балансу нагород (rOwned);

– `ExcludeFromFee(address indexed account, bool isExcludedFromFee)` – подія, що генерується, коли овнер видаляє адресу користувача (account) зі списку для встановлення комісії і змінює логічне значення (isExcludedFromFee);

– `IncludeInFee(address indexed account, bool isExcludedFromFee)` – подія, що генерується, коли овнер додає адресу користувача (account) до списку для встановлення комісії і змінює логічне значення (isExcludedFromFee);

– `TransferFeePercents(uint256 liquidityFee, uint256 taxFee)` – подія, що генерується, коли овнер встановлює відсоток комісії ліквідності (liquidityFee) та податку (taxFee);

– `SwapFeePercents(uint256 liquidityFee, uint256 taxFee)` – подія, що генерується, коли овнер обмінює на протилежні встановлені раніше значення відсотків комісії ліквідності (liquidityFee) та податку (taxFee);

– `MaxTxPercent(uint256 maxTxAmount)` – подія, що генерується, коли овнер встановлює нове значення максимальної суми відсотку при проведенні транзакції (maxTxAmount);

– `ReflectFee(uint256 rTotal, uint256 tFeeTotal)` – подія, що генерується, коли здійснюється транзакція пересилання коштів та змінює відсоткові значення змінних rTotal і tFeeTotal;

– `TakeLiquidity(uint256 rOwned, uint256 tOwned)` – подія, що генерується, коли здійснюється транзакція автоматичного додавання ліквідності при пересиланні коштів за умови досягненні порогового значення та змінює значення змінних rOwned і tOwned балансу смарт-контракту;

– `RemoveAllFee(FeeValues previousSwapFee, FeeValues previousTransferFee, FeeValues swapFee, FeeValues transferFee)` – подія, що генерується, коли здійснюється транзакція

скасування всіх комісій. При цьому змінним swapFee і transferFee встановлюються нульові значення, натомість змінним previousSwapFee і previousTransferFee встановлюються попередні значення комісій;

– `RestoreAllFee(FeeValues swapFee, FeeValues transferFee)` – подія, що генерується, коли здійснюється транзакція відновлення всіх комісій після попереднього скасування. При цьому змінним swapFee і transferFee встановлюються попередні значення комісій;

– `TransferStandard(address indexed sender, address indexed recipient, uint256 rOwnedSender, uint256 rOwnedRecipient)` – подія, що генерується, коли користувач (sender) здійснює пересилання коштів за стандартним типом зі своєї адреси на адресу іншого користувача (recipient). При цьому змінюється значення його поточного балансу токенів (rOwnedSender), а також значення поточного балансу отримувача (rOwnedRecipient);

– `TransferToExcluded(address indexed sender, address indexed recipient, uint256 rOwnedSender, uint256 tOwnedRecipient, uint256 rOwnedRecipient)` – подія, що генерується, коли користувач (sender) здійснює пересилання коштів за стандартним типом зі своєї адреси на адресу іншого користувача (recipient), який входить в список користувачів, комісія для яких є скасованою. При цьому змінюється значення його поточного балансу токенів (rOwnedSender), а також значення поточного балансу отримувача (rOwnedRecipient і tOwnedRecipient);

– `TransferFromExcluded(address indexed sender, address indexed recipient, uint256 rOwnedSender, uint256 tOwnedRecipient, uint256 rOwnedRecipient)` – подія, що генерується, коли користувач (sender), який входить в список користувачів, комісія для яких є скасованою, здійснює пересилання коштів за стандартним типом зі своєї адреси на адресу іншого користувача (recipient). При цьому змінюється значення його поточного балансу токенів (rOwnedSender), а також значення поточного балансу отримувача (rOwnedRecipient і tOwnedRecipient);

– `WithdrawLeftovers(address indexed recipient, uint256 amount)` - подія, що генерується, коли овнер (recipient) знімає залишки BNB з балансу контракту (amount);

– `WithdrawAlienToken(address indexed token, address indexed recipient, uint256 amount)` - подія, що генерується, коли овнер знімає суму (amount) інших токенів (token) з балансу контракту на адресу певного користувача (recipient);

– `ChangeRouter(address indexed router)` – подія, що генерується, коли овнер змінює адресу рутера (router) для доступу до децентралізованої платформи PancakeSwap;

– `AddLiquidity(uint256 amountToken, uint256 amountEth, uint256 liquidity)` - подія, що генерується, коли здійснюється транзакція додавання ліквідності до майнінг пула. При цьому в події зберігаються значення токенів, необхідних для додавання ліквідності (amountToken), сума BNB (amountEth) та сума отриманої ліквідності (liquidity).

2.4 Побудова діаграм послідовності для пересилання і зняття інших токенів

Діаграма послідовності графічно демонструє порядок взаємодії певних об’єктів програми у часі. Як правило, в цій діаграмі демонструється, як користувачі взаємодіють з іншими компонентами програми під час реалізації тих чи інших варіантів використання програми та як при цьому взаємодіють інші компоненти програмної системи [22].

Діаграма послідовності є одним із способів формалізації сценаріїв використання. Її перевага полягає в тому, що на ранніх стадіях опису сценаріїв можливо з’ясувати склад взаємодіючих компонентів та описати потік повідомлень від одних компонентів до інших. Ці компоненти та потоки повідомлень в подальшому будуть трансформовані в конкретні класи (об’єкти), методи цих об’єктів. Відповідно одразу з’ясовується і модель системи подій (Actions), які дані класи (об’єкти) будуть підтримувати та обробляти.

На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси та об’єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній в порядку відправлення.

Для того, щоб здійснити пересилання коштів, необхідно забезпечити таку алгоритмічну послідовність:

– підключити гаманець Metamask до веб-додатку;

– відправити запит на інформацію про баланс користувача, який хоче здійснити пересилання;

– отримати дані;

– визначити, чи не є одна із вказаних адреси адресою овнера;

– порівняти отриману раніше суму балансу користувача зі змінною, яка зберігає значення можливої максимальної суми при транзанкції;

– якщо сума трансферу перевищує дозволену суму для транзакції, слід видати відповідне повідомлення про помилку;

– після того, як транзакція пройшла успішно і баланс контракту більший за суму трансферу, оновити баланс контракту;

– визначити, чи не досягнуто значення порогу для автоматичного додавання ліквідності та обміну токенів і BNB;

– здійснити пересилання, залежно від типу адреси і інформації, чи належить вона до якогось зі списків щодо нарахування нагород чи встановлення комісій;

– відповідно до типу трансферу оновити необхідні змінні;

– отримати відповідь про успішне надсилання коштів;

– оновити баланси гаманця.

У випадку нестачі коштів для виконання транзакції на поточному гаманці транзакція не буде успішною.

Діаграму послідовності для пересилання коштів з контракту чи між користувачами зображено в графічній частині (додаток Б).

Діаграму послідовності для зняття інших токенів власником з балансу смарт-контракту зображено в графічній частині (додаток В).

2.5 Виділення необхідних бібліотек для розроблення смарт-контрактів

Розроблений смарт-контракт токена повинен включати основні необхідні для роботи смарт-контракти наступних бібліотек OpenZeppelin версії 4.6.0, а саме:

– SafeMathUpgradeable – бібліотека математичних функцій для коректних розрахунків, а також для захисту від переповнення комірок надзвичайно великими числами (overflow\underflow) з можливістю оновлення користувацького контракту Arcane-токена;

– OwnableUpgradeable – для керування власністю над токеном, а також надання доступу (контроль безпеки) з можливістю оновлення користувацького контракту Arcane-токена;

– ERC20Upgradeable – бібліотека безпечних функцій типового токена ERC20 для забезпечення безпеки обробки транзакцій з можливістю оновлення користувацького контракту Arcane-токена;

– IERC20 – стандартний інтерфейс токена ERC20, що буде використовуватись для виклику функцій пересилання токенів через екземпляри кожного з токенів у смарт-контракті.

2.6 Розроблення структури програми

Графічний інтерфейс відіграє важливу роль при формуванні загального уявлення про якість програмного продукту. При проєктуванні графічного інтерфейсу слід звернути увагу на такі фактори як ергономіка, зручність роботи і кольорова гамма. Інтерфейс та його якість можуть компенсувати деякі функціональні недоліки додатку, проте бувають випадки, коли інтерфейс невдало підібраний до загального функціоналу додатку.

В процесі конструювання проєкту Arcane було визначено загальну структуру перед програмним розробленням веб-застосунку за допомогою додатку Axure RP 9 Pro Edition.

При проєктуванні графічного інтерфейсу велику увагу приділено ергономіці інтерфейсу, інформативності та загальній зручності роботи. Основною метою є створення інтерфейсу, який буде максимально простий та не викликатиме велике

інформаційне навантаження при роботі з ним.

Щоб почати роботу з веб-застосунком, потрібно під'єднати гаманець з MetaMask. Доки цього не буде зроблено, користувач не зможе повністю взаємодіяти із додатком. Структура даної сторінки представлена на рисунку 2.1.

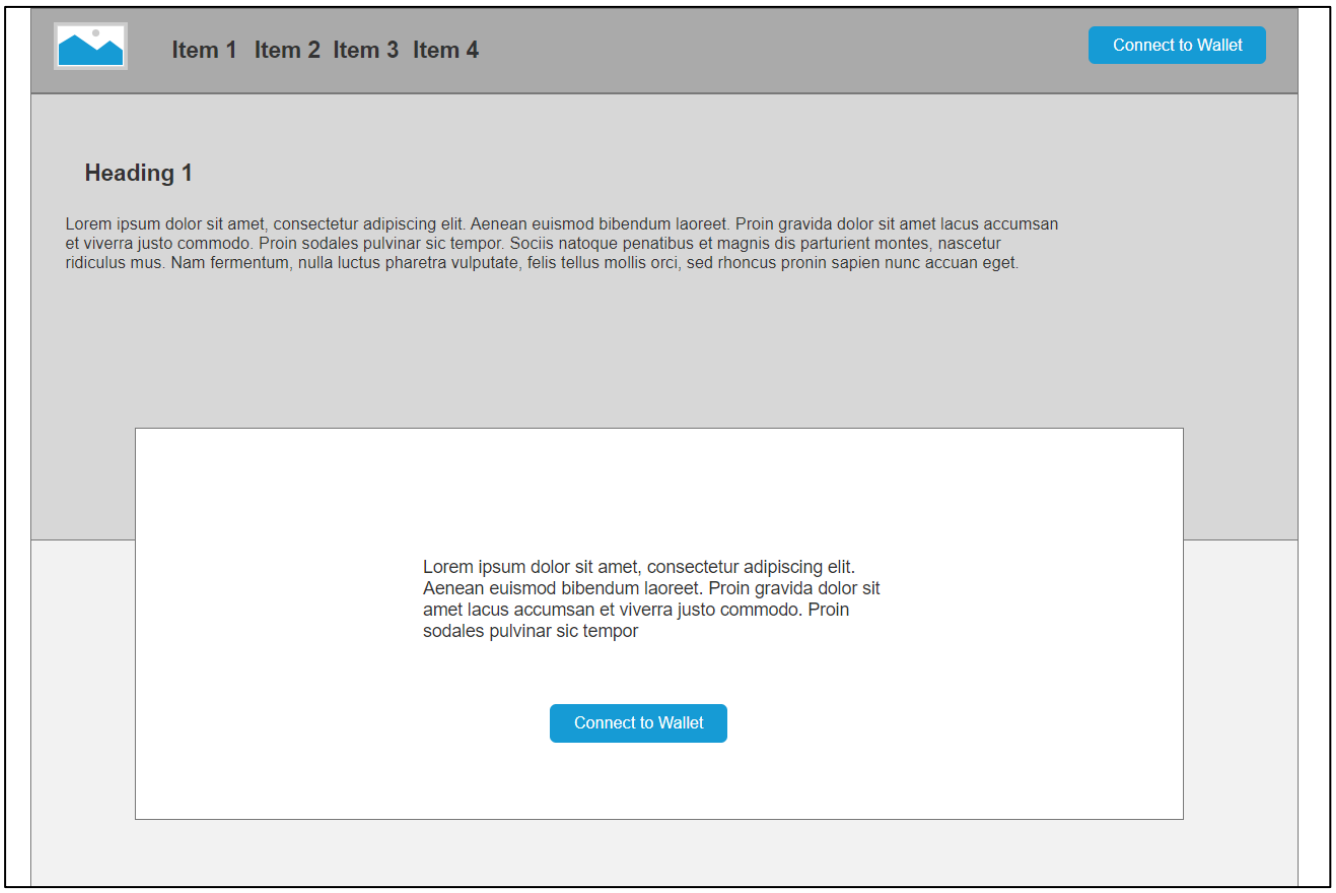

Рисунок 2.1 – Зовнішній вигляд веб-застосунку
перед з'єднанням з MetaMask

Якщо користувач під'єднав гаманець до сайту, в нього з'являється доступ до отримання загальної інформації щодо проєкту Arcane, а також можливість взаємодіяти зі створеними пулами та інтегрованими смарт-контрактами.

Окрім цього, за бажанням користувач може зробити інвестицію в розроблений проєкт шляхом можливого донату, скориставшись для цього послугою, яка описана при завантаженні сторінки в окремому модальному вікні. Структура сторінки після з'єднання гаманця представлена на рисунку 2.2.

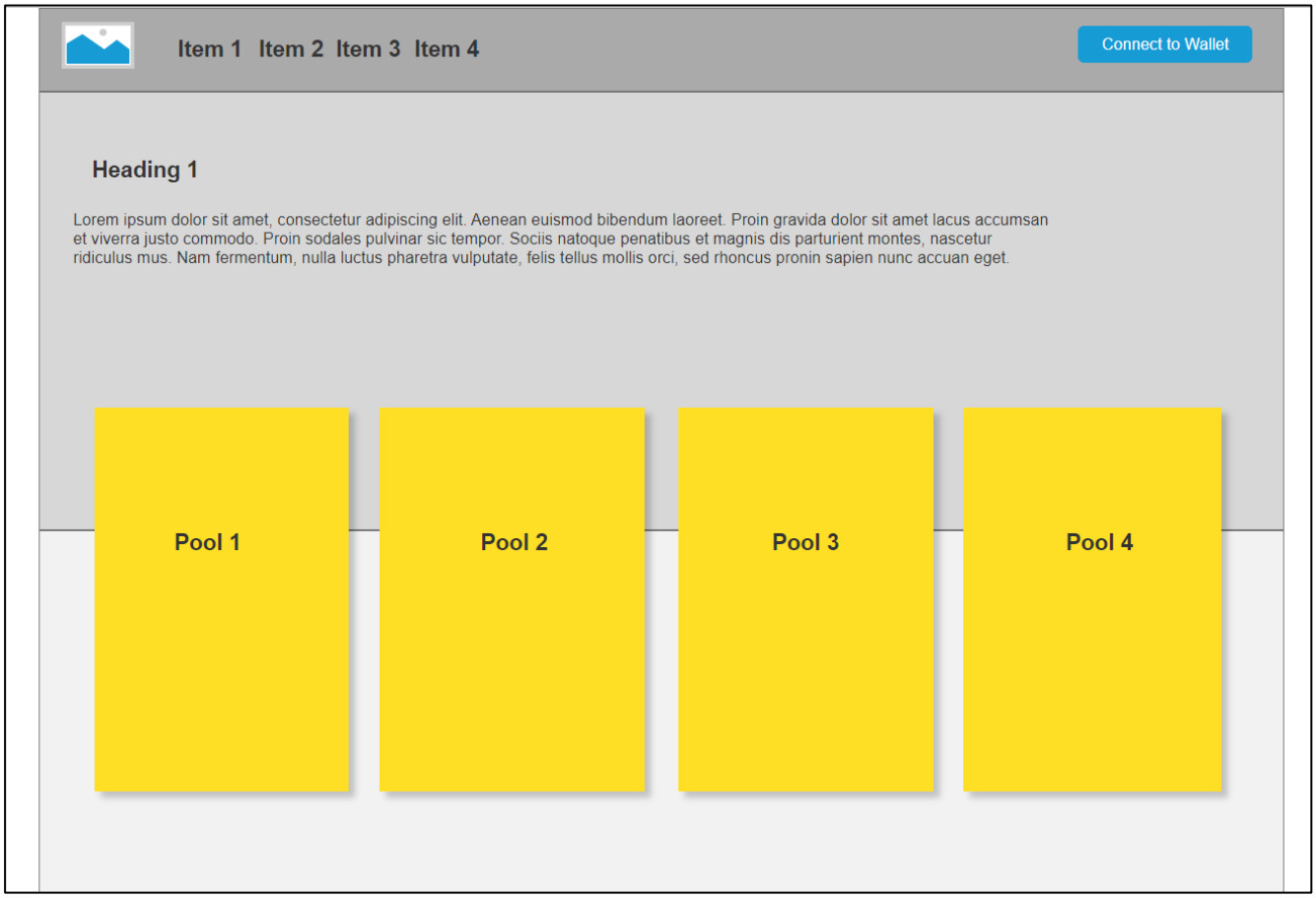

Рисунок 2.2 – Зовнішній вигляд веб-застосунку
після з’єднання з MetaMask

Отже, у даному розділі проведено аналіз та розроблено бізнес-логіку Arcane-токена проєкту, побудовано діаграму вищого рівня та виділено сценарії використання для демонстрації функціоналу смарт-контрактів, розроблено діаграми послідовності для пересилання токенів за допомогою типових засобів та методів блокчейну. Окрім цього, було спроєктовано загальну структуру веб-застосунку, з яким буде взаємодіяти користувач при різних варіантах подій.

3 ПРОГРАМНА ЧАСТИНА

3.1 Написання смарт-контракту токена

Смарт-контракти – це програмні контракти, завантажені в децентралізовану мережу Ethereum, які дозволяють двом контрагентам встановлювати умови транзакції без необхідності довіряти її виконання якійсь третій стороні – посереднику. Об'єднані контракти, які виконують надскладні процеси і розрахунки, складають децентралізований додаток [23].

Окрім цього, вони надають можливість безпечно здійснювати фінансові операції, використовуючи криптовалюти, гроші чи певні цінні папери.

Елементами «розумного» контракту є:

– сторони угоди, що мають цифровий підпис, які погоджуються або відмовляються від відповідності товару або послуги висунутим раніше вимогам;

– предмет договору – товар або послуги, які будуть відправлені в обмін на грошові кошти;

– умови, при дотриманні яких буде проведений автоматичний обмін благами, наприклад, відповідність поставленого товару стандартам якості;

– децентралізована платформа, в якій написаний алгоритм (програмний код) самого смарт-контракту.

Особливість смарт-контрактів в тому, що завдяки складовим елементам вони значно економлять час і ресурси, мають низькі витрати, надають додаткову безпеку через розміщення на блокчейні, а також швидко перевіряють умови виконання контрактів.

При розробленні смарт-контракту ArcaneToken було враховано та недопущено будь-які ризики втрати коштів, проведено рефакторинг з метою зменшення оплати за транзакції, ціна яких залежить від кількості та обсягу виконаних операцій, а також описано NatSpec-документацію перед кожним елементом у договорі.

Приклад реалізації методу для пересилання токенів між адресами користувачів:

```
/// @notice Transfer amount, add liquidity
/// @dev Check amount and accounts, transfer will take fee, add liquidity
/// @param from Address of account that transfer amount
/// @param to Address of account that get transfer's amount
/// @param amount Value of amount for transfer
function _transfer(
   address from,
   address to,
   uint256 amount
) internal virtual override {
   require(from != address(0), "ERC20: transfer from the zero address");
   require(to != address(0), "ERC20: transfer to the zero address");
   require(amount > 0, "Transfer amount must be greater than zero");
   if (from != owner() && to != owner())
     require(
       amount <= maxTxAmount,
       "Transfer amount exceeds the maxTxAmount."
   );

   uint256 contractTokenBalance = balanceOf(address(this));
   if (contractTokenBalance >= maxTxAmount) {
       contractTokenBalance = maxTxAmount;
   }
   bool overMinTokenBalance = contractTokenBalance >=
       _numTokensSellToAddToLiquidity;
   if (
       overMinTokenBalance &&
       !_inSwapAndLiquify &&
       from != uniswapV2Pair &&
       swapAndLiquifyEnabled
    ) {
       contractTokenBalance = _numTokensSellToAddToLiquidity;
       _swapAndLiquify(contractTokenBalance);
    }
    bool takeFee = true;
    if (_isExcludedFromFee[from] || _isExcludedFromFee[to]) {
       takeFee = false;
    }
    _tokenTransfer(from, to, amount, takeFee);
}
```

Лістинг повного коду смарт-контракту ArcaneToken наведено в додатку А.

3.2 Розгортання смарт-контракту на тестовій мережі

Для розгортання смарт-контракту обрано тестову мережу TESTNET Binance (BscScan). Розгортання здійснюється з можливістю викликати будь-яку функцію та перевірити коректність її роботи перед офіційним релізом в основній мережі, сплативши комісію за допомогою тестової рідної (нативної) валюти BNB. Усі токени, створені на BSC Testnet, не мають реальної цінності, а використовуються лише в цілях тестування готового проєкту.

Щоб успішно розгорнути контракти на мережі, необхідно виконати наступні кроки:

– встановити модуль Truffle;

– ініціалізувати проєкт для цього середовища;

– встановити постачальник гаманців;

– відкомпілювати попередньо створені контракти з метою отримання їх конструкцій;

– написати в конфігураційних файлах та міграційних скриптах сценарії розгортання контрактів, що представляють собою лістинг коду покрокового асинхронного налаштування;

– задати налаштування мережі: номер, rpc-посилання на один з вузлів мережі, приватний ключ гаманця, власник якого вважатиметься деплоєром, максимальна кількість газу, виділена для однієї транзакції, значення ціни за витрачену одиницю газу, а також кількість підтверджень після отримання транзакції в блокчейні, кількість тайм-аут блоків, що допускаються бути пропущеними тощо;

– запустити сценарії розгортання на виконання за допомогою команди:

```
truffle migrate --network назва_мережі;
```

– перевірити та зберегти результати розгортання;

– верифікувати смарт-контракти за допомогою попередньо скомпільованого abi, байт-коду та наступної команди:

```
truffle run verify назва_контракту@адреса
```

Приклад налаштування параметрів мережі:

```
bsc_testnet: {
      provider: () => new HDWalletProvider([process.env.PRIVATE_KEY],
`https://data-seed-prebsc-1-s1.binance.org:8545`),
      network_id: 97,
      confirmations: 2,
      timeoutBlocks: 200,
      gas: 5000000,
      gasPrice: 30000000000,
}
```

Приклад сценарію розгортання головного токена веб-додатку:

```
const { deployProxy } = require('@openzeppelin/truffle-upgrades');

const ArcaneToken = artifacts.require('ArcaneToken');

module.exports = async function (deployer) {
const router = "0xD99D1c33F9fC3444f8101754aBC46c52416550D1"; // router address for
bscscan testnet
const owner = "0x9b38DE554AC02af25c911e262690550c8584BFaa"; // address where all
tokens and ownership will be transfered
const instance = await deployProxy(ArcaneToken, [router, owner], { deployer,
initializer: 'initialize' });
console.log('Deployed', instance.address);
```

```
};
```

Лістинг коду сценаріїв розгортання усіх смарт-контрактів проєкту наведено в додатку Б.

3.3 Реалізація модульних тестів

Модульне тестування – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою.

Модульні тести, або unit-тести, розробляють в процесі розробки програмісти та, іноді, тестувальники білої скриньки (white-box testers).

Зазвичай unit-тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку [24].

Тестування програмного забезпечення не може знайти всіх помилок у програмі. У більшості програм неможливо прорахувати кожен варіант виконання. Це також стосується модульного тестування. Крім того, модульне тестування повинно тестувати тільки модулі. Тому даний вид тестування не зможе знайти інтеграційні помилки та інші: помилки архітектури, проблеми з витримкою навантажень на ПЗ. Unit-тестування має проводитись разом з іншими видами тестування програмного забезпечення. Як і будь-який вид тестування, модульне тестування може визначити лише наявність помилок, а не їх відсутність.

Приклад тестових випадків для зняття залишку BNB з наявного балансу:

```
describe("ArcaneToken Withdraw Functions Phase Test Cases", function () {

        it("should withdraw leftovers correctly", async () => {
            const amountPairE = ether("10000");
            const amountPairW = ether("10");
            const amountT1 = ether("1000");
            const amountT2 = ether("100");
            const amountT3 = ether("10");
            const fee = new BN("0");
            const feeT = new BN("0");
            const deadline = (await time.latest()).add(new BN("100000"));
            const newRouter = await IUniswapV2Router02.at(ROUTER_ADDRESS);
            const weth = await newRouter.WETH();
            const newWeth = await IWETH.at(weth);

            let currentRate = _rTotal.div(_tTotal);
            let tFeeT1 = amountT1.mul(fee).div(new BN("100"));
            let tLiquidityT1 = amountT1.mul(fee).div(new BN("100"));
            let rAmountT1 = amountT1.mul(currentRate);
            let rFeeT1 = tFeeT1.mul(currentRate);
            let rLiquidityT1 = tLiquidityT1.mul(currentRate);
```

```
            let rTransferAmountT1 = rAmountT1.sub(rFeeT1).sub(rLiquidityT1);
            let rOwnedRT1 = rTransferAmountT1;
            _rTotal = _rTotal.sub(rFeeT1);
            currentRate = _rTotal.div(_tTotal);

            let tFeeT2 = amountT2.mul(feeT).div(new BN("100"));
            let tLiquidityT2 = amountT2.mul(feeT).div(new BN("100"));
            let rAmountT2 = amountT2.mul(currentRate);
            let rFeeT2 = tFeeT2.mul(currentRate);
            let rLiquidityT2 = tLiquidityT2.mul(currentRate);
            let rTransferAmountT2 = rAmountT2.sub(rFeeT2).sub(rLiquidityT2);
            let rOwnedST2 = rOwnedRT1.sub(rAmountT2);
            let rOwnedRT2 = rTransferAmountT2;
            _rTotal = _rTotal.sub(rFeeT2);
            currentRate = _rTotal.div(_tTotal);

            let tFeeT3 = amountT3.mul(feeT).div(new BN("100"));
            let tLiquidityT3 = amountT3.mul(feeT).div(new BN("100"));
            let rAmountT3 = amountT3.mul(currentRate);
            let rFeeT3 = tFeeT3.mul(currentRate);
            let rLiquidityT3 = tLiquidityT3.mul(currentRate);
            let rTransferAmountT3 = rAmountT3.sub(rFeeT3).sub(rLiquidityT3);
            let rOwnedST3 = rOwnedRT2.sub(rAmountT3);
            let rOwnedRT3 = rOwnedST2.add(rTransferAmountT3);
            _rTotal = _rTotal.sub(rFeeT3);
            currentRate = _rTotal.div(_tTotal);
            let balanceUser2T3 = rOwnedST3.div(currentRate);
            let balanceUser1T3 = rOwnedRT3.div(currentRate);
            await arcane.setTransferFeePercent(fee, fee);
            await newWeth.deposit({ value: amountPairW, gas: 5500000 });
            await newWeth.approve(newRouter.address, amountPairW);
            await arcane.approve(newRouter.address, amountPairE);
            await arcane.setThreshold(ether("0.01"));
                   await  newRouter.addLiquidity(arcane.address,  newWeth.address,
amountPairE, amountPairW, 0, 0, owner, deadline);
            await arcane.transfer(user1, amountT1);
            await arcane.transfer(user2, amountT2, { from: user1 });
            await arcane.transfer(user1, amountT3, { from: user2 });
                                                                        (await
arcane.balanceOf(user2)).should.be.bignumber.equal(balanceUser2T3);
                                                                        (await
arcane.balanceOf(user1)).should.be.bignumber.equal(balanceUser1T3);
            let beforeBalanceC = await balance.current(arcane.address);
            let result = await arcane.withdrawLeftovers();
            expectEvent(
                result,
                "WithdrawLeftovers",
                { recipient: owner, amount: beforeBalanceC }
            );
              (await balance.current(arcane.address)).should.be.bignumber.equal(new
BN("0"));
        });

    });
```

Приклад тестових випадків для інших методів зняття коштів з наявного балансу:

```
it("should withdraw alien tokens correctly", async () => {
   const mockToken = await MockERC20.new(ether("100"));
   await mockToken.mint(arcane.address, ether("100"));
   let result = await arcane.withdrawAlienToken(mockToken.address, user1,
ether("20"));
   expectEvent(
```

```
        result,
        "WithdrawAlienToken",
        { token: mockToken.address, recipient: user1, amount: ether("20")
    })
    (await mockToken.balanceOf(user1)).should.be.bignumber.equal(ether("20"));
});

it("shouldn't withdraw alien tokens if token's address is arcane and swap&liquify
if enabled", async () => {
    await expectRevert(
        arcane.withdrawAlienToken(arcane.address, user1, ether("20")),
        "Token can not be ARC"
    );
});

it("should withdraw alien tokens by the owner if token's address is arcane and
swap&liquify if disabled", async () => {
    const amount = ether("1")
    await arcane.transfer(arcane.address, amount);
    (await arcane.balanceOf(arcane.address)).should.be.bignumber.equal(amount);
    await arcane.setSwapAndLiquifyEnabled(false);
    await arcane.withdrawAlienToken(arcane.address, user1, amount);
    (await arcane.balanceOf(arcane.address)).should.be.bignumber.equal("0");
});

it("shouldn't withdraw alien tokens by not the owner if token's address is arcane
and swap&liquify if disabled", async () => {
    const amount = ether("1")
    await arcane.transfer(arcane.address, amount);
    (await arcane.balanceOf(arcane.address)).should.be.bignumber.equal(amount);
    await arcane.setSwapAndLiquifyEnabled(false);
    await expectRevert(
        arcane.withdrawAlienToken(arcane.address, user1, ether("1"), { from: user1
      }),
        "Ownable: caller is not the owner"
    );
});

it("shouldn't withdraw alien tokens if amount is zero", async () => {
    const mockToken = await MockERC20.new(ether("100"));
    await mockToken.mint(arcane.address, ether("10"));
    await expectRevert(
       arcane.withdrawAlienToken(mockToken.address, user1, ether("13")),
       "Insufficient tokens balance"
    );
});
```

Лістинг коду тестування усіх методів смарт-контракту ArcaneToken наведено в додатку В.

3.4 Реалізація функціоналу front-end частини

Графічний інтерфейс дипломного проєкту побудований з використанням фреймворка Angular.

До основних переваг фреймворка Angular належать: декларативний стиль коду; використання директив; висока швидкість розробки; SPA (Single page application); модульність; наявність готових рішень; простота тестування.

Окрім цього, для побудови деяких елементів використано засоби бібліотеки Applicature Univarsal Components.

Зовнішній вигляд повністю сформованого додатку зображено на рисунку 3.1.

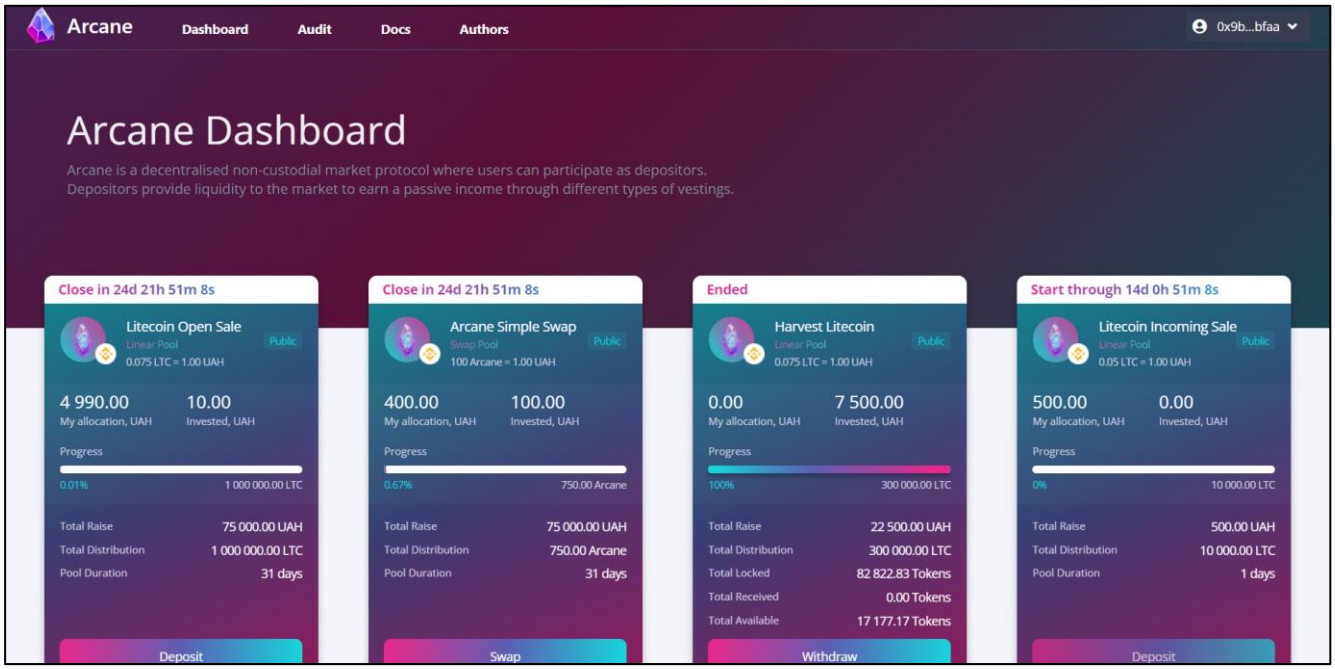

Рисунок 3.1 – Загальний вигляд веб-застосунку при завантаженні

Лістинг коду методу завантаження застосунку:

```
public async ngOnInit(): Promise<void> {

    if (window.localStorage.getItem('account') == null ||
window.localStorage.getItem('account') == 'undefined') {
      (<HTMLInputElement>document.getElementById('pools')).style.display = 'none';
      (<HTMLInputElement>document.getElementById('user-icon')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('arrow-up')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('arrow-down')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('dropdown-
content')).style.display = 'none';
      (<HTMLInputElement>document.getElementById('how-it-
works__section')).style.display = 'none';
      (<HTMLInputElement>document.getElementById('future-section')).style.display
= 'none';
      (<HTMLInputElement>document.getElementById('team__container')).style.display
= 'none';
      (<HTMLInputElement>document.getElementById('secure-section')).style.display
= 'none';
      (<HTMLInputElement>document.getElementById('pre-footer')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('footer')).style.display =
'none';
    } else {
      let account = window.localStorage.getItem('account');
      let firstLetters = account.substring(0, 4);
      let lastLetters = account.substring(38, 42);
```

```
      let subAddress = firstLetters + '...' + lastLetters;
      (<HTMLInputElement>document.getElementById('load-connect-
wallet')).style.display = 'none';
      (<HTMLInputElement>document.getElementById("connect-wallet-
button")).innerText = subAddress;
      (<HTMLInputElement>document.getElementById("connect-wallet-
button")).style.background = '#383d51';
      (<HTMLInputElement>document.getElementById("connect-wallet-
button")).style.paddingLeft = '35px';
      (<HTMLInputElement>document.getElementById("connect-wallet-
button")).style.paddingRight = '35px';
      (<HTMLInputElement>document.getElementById('arrow-up')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('dropdown-
content')).style.display = 'none';

      (<HTMLInputElement>document.getElementById('pools')).style.display = 'flex';
      (<HTMLInputElement>document.getElementById('user-icon')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('arrow-down')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('future-section')).style.display
= 'block';
      (<HTMLInputElement>document.getElementById('how-it-
works__section')).style.display = 'block';
      (<HTMLInputElement>document.getElementById('team__container')).style.display
= 'block';
      (<HTMLInputElement>document.getElementById('secure-section')).style.display
= 'block';
      (<HTMLInputElement>document.getElementById('pre-footer')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('footer')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('donate-window')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('donate-window')).style.animation
= 'slide-left 0.7s';

      this.initContracts();
    }

    addEventListener('click', function handleClick(this: HTMLElement) {
      (<HTMLInputElement>document.getElementById('arrow-up')).style.display =
'none';
      (<HTMLInputElement>document.getElementById('arrow-down')).style.display =
'block';
      (<HTMLInputElement>document.getElementById('dropdown-
content')).style.display = 'none';
    });

    window.ethereum.on('chainChanged', () => {
      window.location.reload();
    });

    window.ethereum.on('accountsChanged', () => {
      window.location.reload();
    });
  }
```

Лістинг HTML-, SCSS- та TS-коду для реалізації front-end частини наведено в додатку Г.

3.5 Проведення мануальної частини додаткового тестування

При мануальній частині ручного тестування тестувальники вручну виконують тести, не використовуючи ніяких засобів автоматизації. Ручне тестування – низькорівневий та простий тип тестування, що не вимагає великої кількості додаткових знань.

Тим не менш, перед тим як автоматизувати тестування будь-якого додатку, необхідно спочатку виконати серію тестів вручну. Мануальне тестування вимагає більших зусиль, але без нього неможливо переконатися в тому, чи можлива автоматизація взагалі. Один із фундаментальних принципів тестування свідчить: 100% автоматизація неможлива, тому ручне тестування – це необхідність [25].

Приклад опису знайденого бага при мануальному тестуванні:

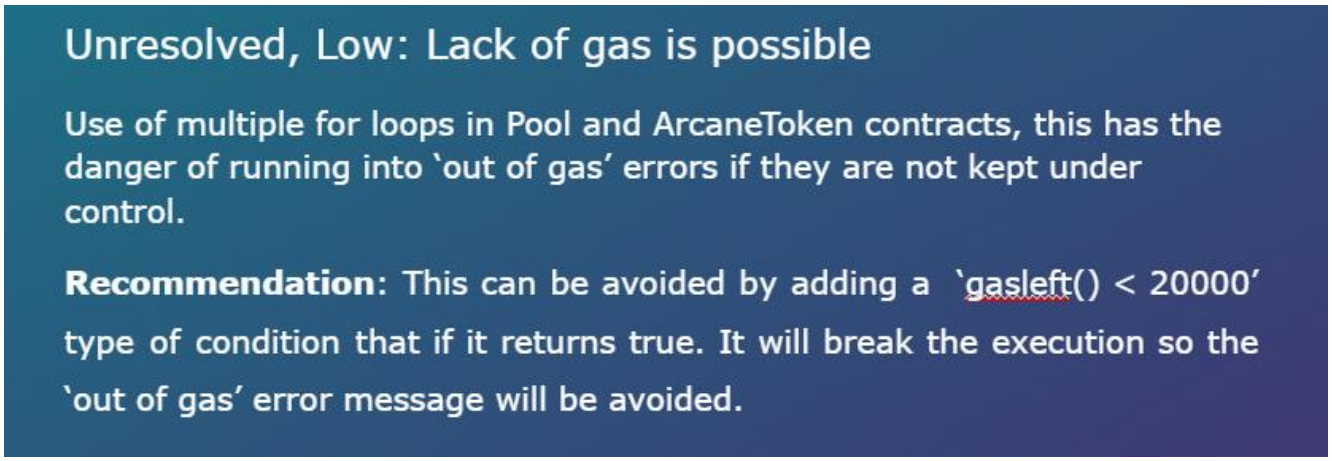

Рисунок 3.2 – Опис знайденої помилки при проведенні аудиту

4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1 Основні принципи першочергового налаштування

Для коректної взаємодії із створеним веб-застосунком необхідно зробити деякі першочергові налаштування. Оскільки провідним елементом проєкту Arcane, через який здійснюється вся взаємодія з Dapps, є гаманець з додатку MetaMask, для початку необхідно встановити це розширення у своєму браузері (рисунок 4.1):

Рисунок 4.1 – Розширення для збереження гаманців

Після завантаження розширення MetaMask необхідно встановити пароль, за допомогою якого буде захищено доступ до усіх локальних гаманців браузера (рисунок 4.2):

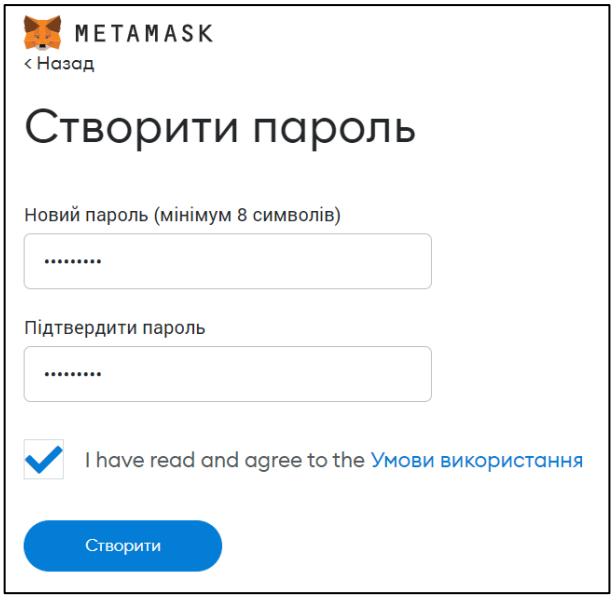

Рисунок 4.2 – Форма для введення паролю

Наступним етапом є надання секретної резервної фрази новоствореного гаманця (рисунок 4.3), що являє собою фразу з 12 слів і є ключем до гаманця та його коштів. Її необхідно обов'язково зберегти в захищеному сховищі.

Рисунок 4.3 – Приклад резервної фрази

Після підтвердження фрази для відновлення MetaMask надасть порожній гаманець (рисунок 4.4), в якому можна зберігати різнотипну криптовалюту.

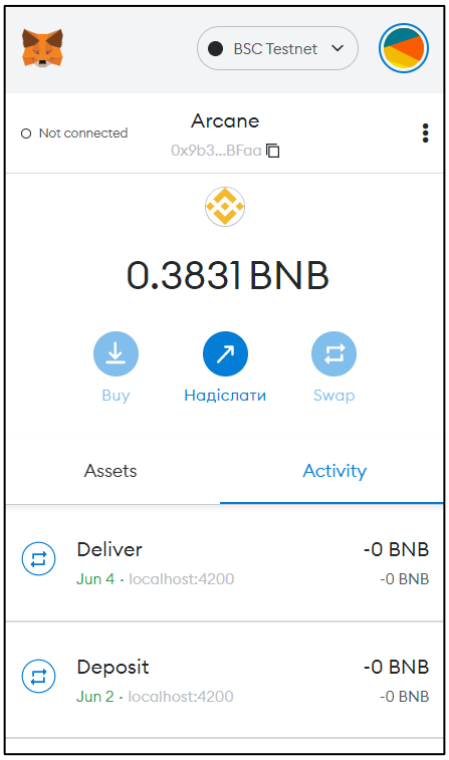

Рисунок 4.4 – Приклад створеного акаунту

Щоб запустити front-end частину, яка буде розгорнута за адресою localhost:4200, необхідно скористатися наступною командою: ng serve -o.

4.2 Результати виконання модульних тестів

Результати виконання модульних тестів для смарт-контракту ArcaneToken:

```
Contract: ArcaneToken
   ArcaneToken Initializing Phase Test Cases
     ✓ should set router correctly (142ms)
     ✓ should get balance of owner correctly (209ms)
     ✓ should exclude from fee correctly (263ms)
     ✓ should set _maxTxAmount correctly (171ms)
     ✓ should set swap fee correctly (251ms)
     ✓ should set transfer fee correctly (240ms)
     ✓ should set swapAndLiquifyEnabled correctly (109ms)
     ✓ should set token's name correctly (148ms)
     ✓ should set token's symbol correctly (169ms)
     ✓ should set total supply correctly (142ms)
     ✓ should set decimals correctly (112ms)
   ArcaneToken Get/Set Functions Phase Test Cases
     ✓ should set threshold correctly (502ms)
     ✓ should deliver correctly (817ms)
     ✓ shouldn't deliver if account isn't excluded (1470ms)
     ✓ should exclude account from reward without _rOwned correctly (1024ms)
     ✓ should exclude account from reward with _rOwned correctly (1416ms)
     ✓ shouldn't exclude account from reward if account isn't excluded (926ms)
     ✓ should include account in reward correctly (1991ms)
     ✓ shouldn't include account in reward if account is already excluded (364ms)
     ✓ should set transfer fee percent correctly (818ms)
     ✓ shouldn't set tax fee percent if value more than 100 (912ms)
     ✓ shouldn't set tax fee percent if caller isn't owner (413ms)
     ✓ should set swap fee percent correctly (691ms)
     ✓ shouldn't set liquidity fee percent if value more than 100 (640ms)
     ✓ should set max tx percent correctly (623ms)
     ✓ shouldn't set max tx percent if value more than 100 (471ms)
     ✓ should update router correctly (441ms)
     ✓ shouldn't update router if zero's address (347ms)
     ✓ should exclude account from fee correctly (426ms)
     ✓ should include account in fee correctly (566ms)
     ✓ should set enable for swap and liquify correctly (459ms)
     ✓ should lock correctly (1567ms)
     ✓ should unlock correctly (1104ms)
     ✓ shouldn't unlock if caller haven't permission (413ms)
     ✓ shouldn't unlock if _lockTime isn't exceeds (964ms)
     ✓ should return reflection from token correctly (337ms)
     ✓ shouldn't return reflection from token if amount more than supply (119ms)
   ArcaneToken Transfer Functions Phase Test Cases
     ✓ shouldn't transfer amount of tokens if insufficient allowance (743ms)
     ✓ shouldn't transfer amount of tokens if zero's 'to' address (349ms)
     ✓ shouldn't transfer amount of tokens if amount is zero (423ms)
     ✓ shouldn't transfer amount of tokens if amount exceeds the maxTxAmount
(516ms)
     ✓ should transfer amount of tokens from excluded account correctly (3578ms)
     ✓ should transfer amount of tokens from excluded account correctly if
contract's address is excluded (4056ms)
     ✓ should transfer amount of tokens to excluded account correctly (3278ms)
```

```
      ✓ should transfer amount of tokens correctly if both accounts is excluded
(4327ms)
      ✓ should standard transfer amount of tokens correctly if not satisfying the
threshold (2699ms)
      ✓ should standard transfer amount of tokens correctly without fee (3613ms)
      ✓ should standard transfer amount of tokens correctly if not satisfying the
threshold and with fee (2141ms)
      ✓ should standard transfer amount of tokens correctly if satisfying the
threshold (8825ms)
    ArcaneToken Withdraw Functions Phase Test Cases
      ✓ should withdraw leftovers correctly (9383ms)
      ✓ should withdraw alien tokens correctly (4632ms)
      ✓ shouldn't withdraw alien tokens if token's address is arcane and
swap&liquify if enabled (553ms)
      ✓ should withdraw alien tokens by the owner if token's address is arcane and
swap&liquify if disabled (4129ms)
      ✓ shouldn't withdraw alien tokens by not the owner if token's address is
arcane and swap&liquify if disabled (2791ms)
      ✓ shouldn't withdraw alien tokens if amount is zero (1573ms)
      ✓ shouldn't withdraw alien tokens if amount is zero (1870ms)
    ArcaneToken Fee Phase Test Cases
      ✓ should transfer tokens depending on transfer fee correctly taxFee = 0,
liquidityFee = 2 (3478ms)
      ✓ should transfer tokens depending on fee correctly taxFee = 50, liquidityFee
= 0 (4247ms)
      ✓ should transfer tokens depending on the buy fee correctly (8604ms)
      ✓ should transfer tokens correctly depending on the fee while add liquidity
      ✓ should transfer tokens depending on the sell fee correctly
      ✓ Balances after transfer all tokens when _taxFee = 0; _liquidityFee = 2

  62 passing (3m21s)
```

4.3 Результати роботи на тестових даних

Для перевірки правильності виконання поставлених задач, а також функціональних можливостей програмного продукту, важливим етапом є виконання програми з тестовими даними.

Для початку взаємодії з веб-додатком необхідно під’єднати до нього обраний гаманець на певній мережі з MetaMask – розширення браузера. Процес з’єднання гаманця MetaMask з Dapps представлено на рисунках 4.5 – 4.6.

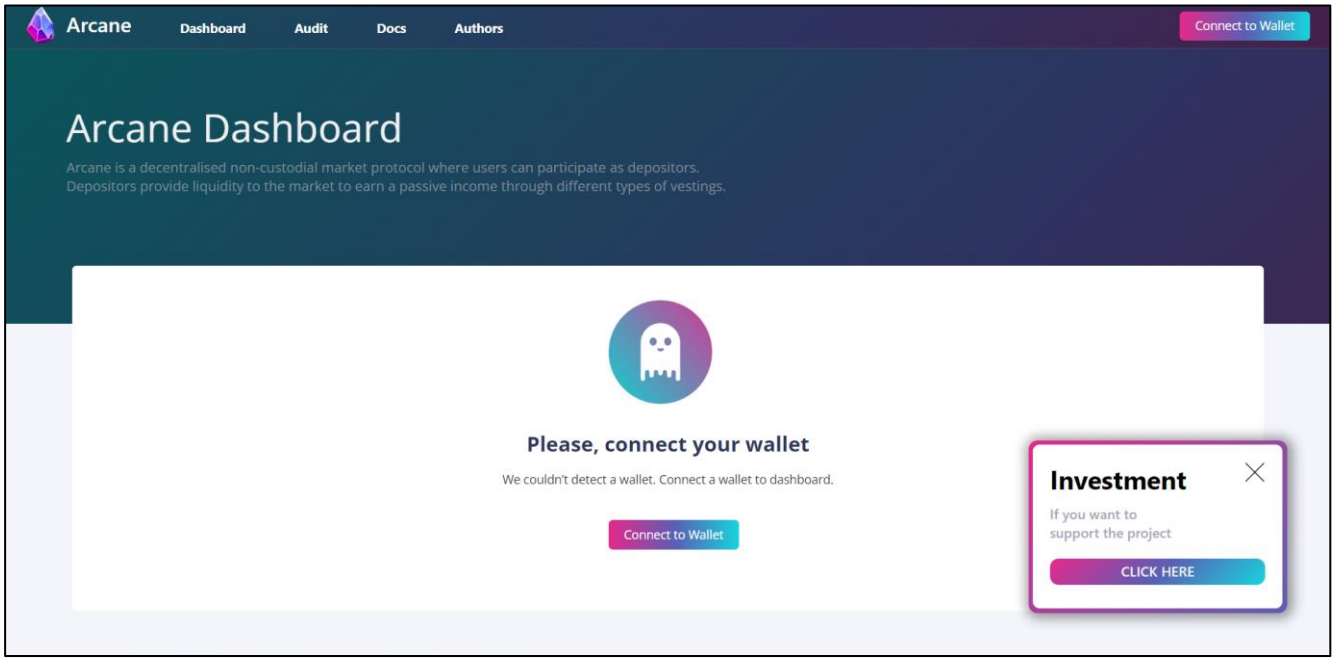

Рисунок 4.5 – Сторінка застосунку при завантаженні

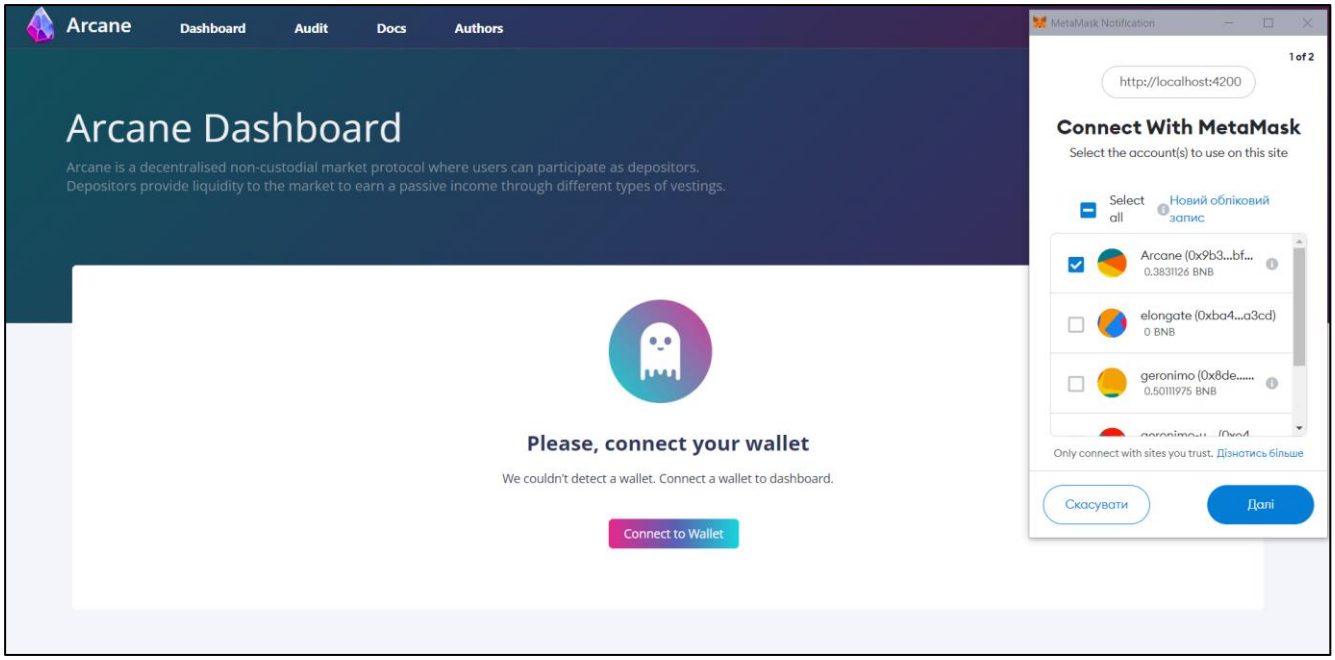

Рисунок 4.6 – Вибір мережі та гаманця для підключення

Після вибору мережі та потрібного гаманця необхідно підтвердити в браузерному розширенні MetaMask бажання під’єднатися до децентралізованого додатку (рисунок 4.7):

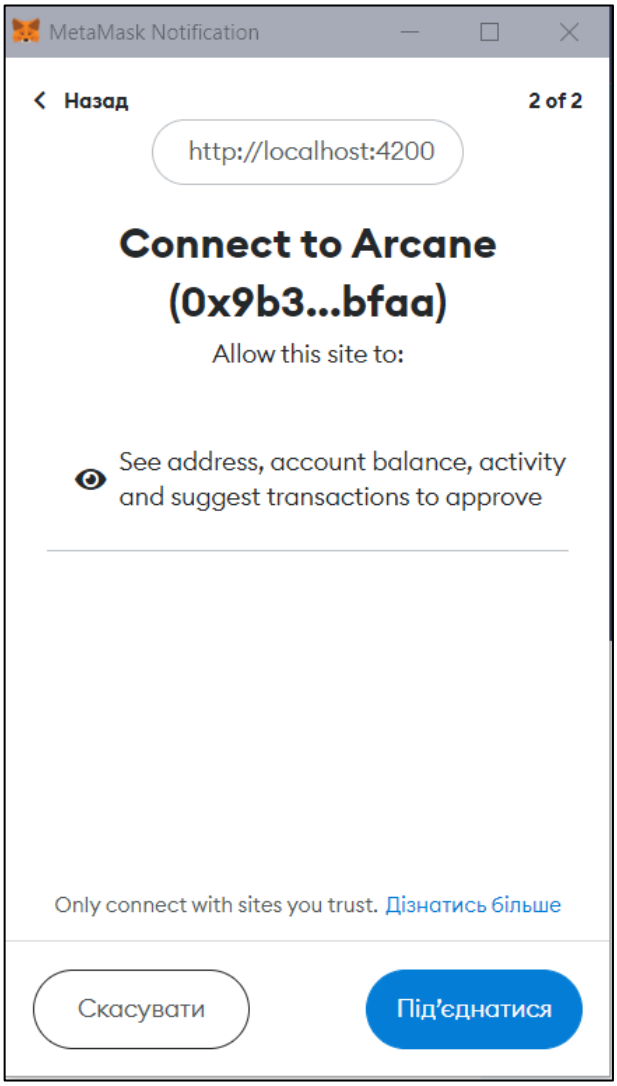

Рисунок 4.7 – Під'єднання гаманця до застосунку

Після успішного з'єднання обраного гаманця користувача із застосунком, завантажується сайт з доступною інформацією для кожного, хто під'єднався, щодо проєкту Arcane, його авторів, технічних документацій, графіків та проведеного аудиту. Окрім цього, користувачу відображаються дані пулів, в яких він робив депозит, на основі Arcane-токену та вестінгів. На рисунку 4.8 зображено завантажену сторінку після успішного під'єднання гаманця користувача:

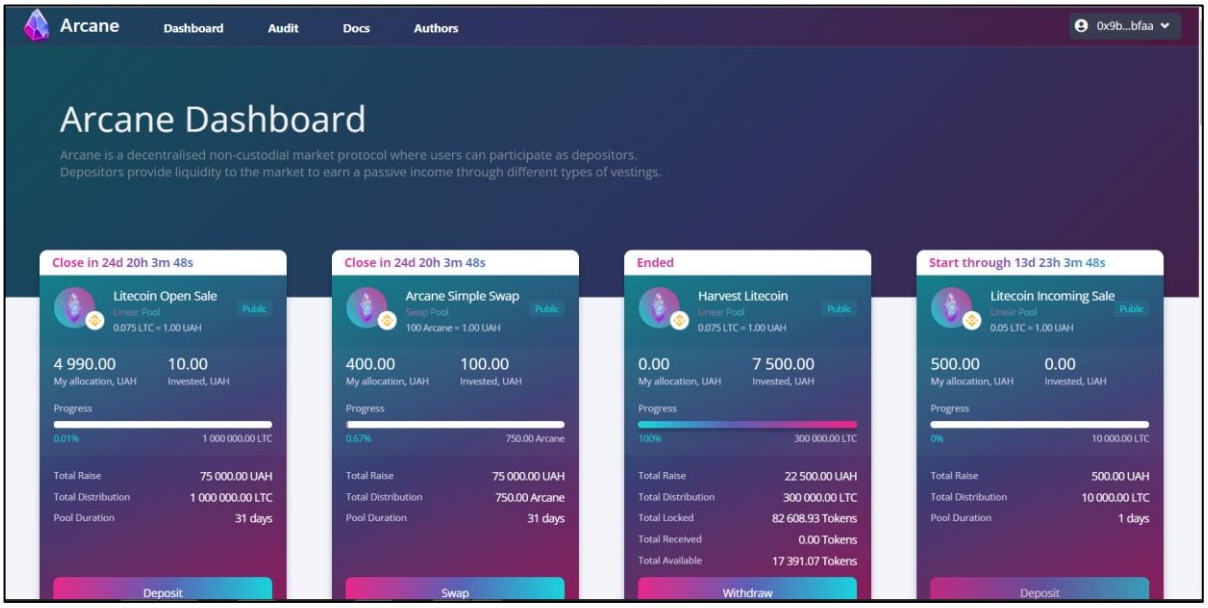

Рисунок 4.8 – Стартова сторінка після під’єднання гаманця

На завантаженій сторінці вгорі розташований хедер (шапка сайту), на якому вказано меню з посилання на сформовані сторонні джерела (створені командою Arcane технічні документації, графіки та додаткові пояснення), а також розташована кнопка з відображенням частини адреси під’єднаного гаманця. При кліку на кнопку відображається dropdown-меню, де користувач може дізнатися детальнішу інформацію про мережу, а також скопіювати адресу власного гаманця за один клік чи від’єднатись від даного додатку (рисунок 4.9):

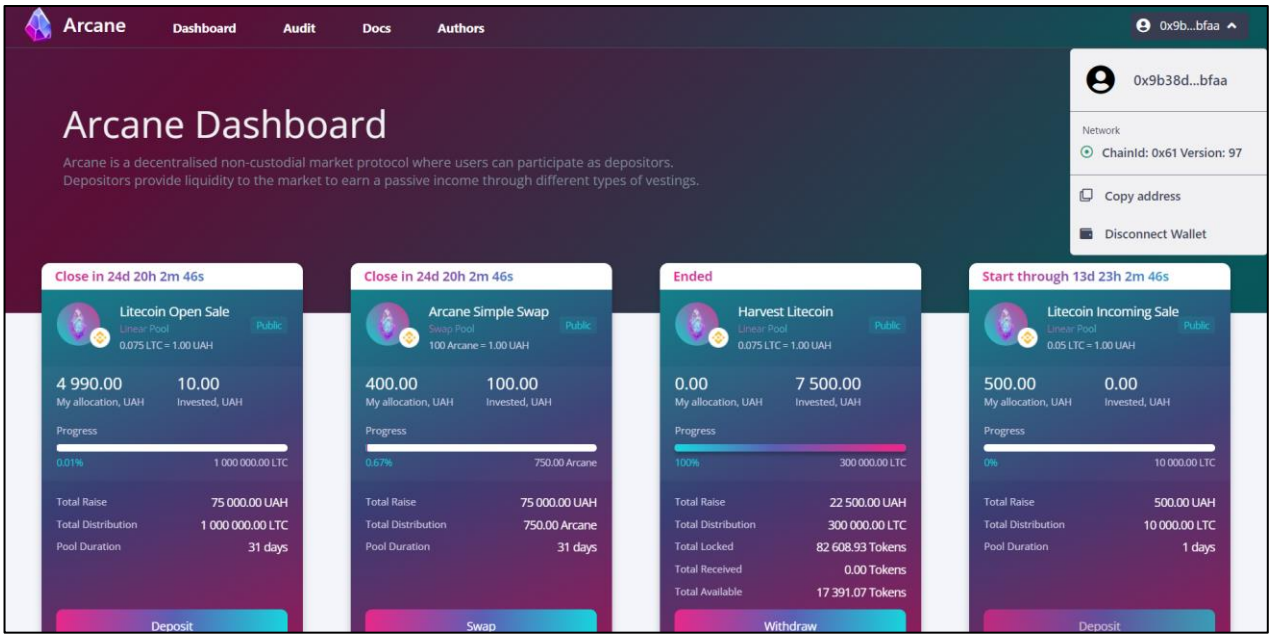

Рисунок 4.9 – Відображення додаткового меню програми

Якщо користувач бажає здійснювати певні можливі дії зі створеними пулами, спочатку йому потрібно схвалити суму, яку він хоче використати з власного балансу. Для успішної реалізації цієї дії відображається модальне вікно, де користувач має можливість ввести суму токенів, які буде використовувати, схвалити їх за допомогою функції approve() зі смарт-контракту токена, заплатити комісію за транзакцію в MetaMask. У випадку успішного проходження транзакції можна зробити депозит чи обміняти токени, виконуючи 2 крок в модальному вікні (рисунки 4.10–4.11):

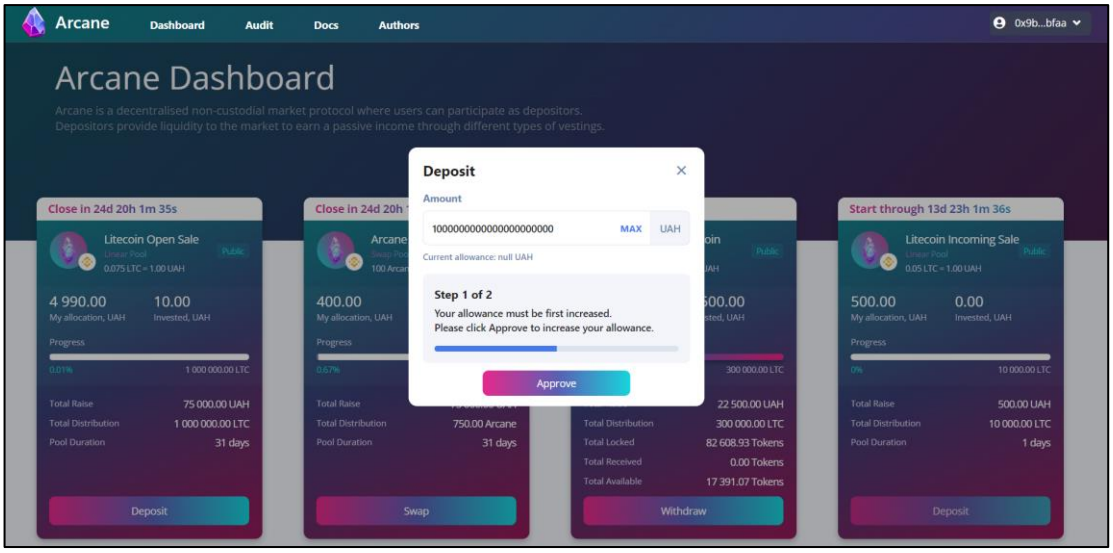

Рисунок 4.10 – Схвалення суми токенів для використання (крок 1)

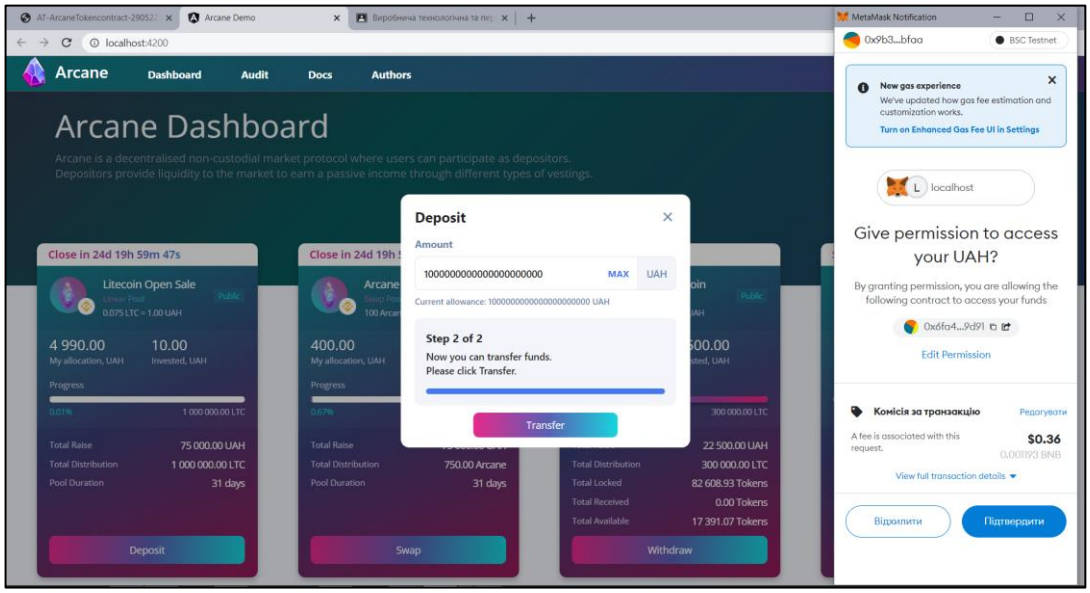

Рисунок 4.11 – Реалізація депозиту токенів в пул з відкритим продажем (крок 2)

Після здійснення вищенаведених дій користувач отримує сповіщення про успішну транзакцію в розширенні MetaMask (рисунок 4.12):

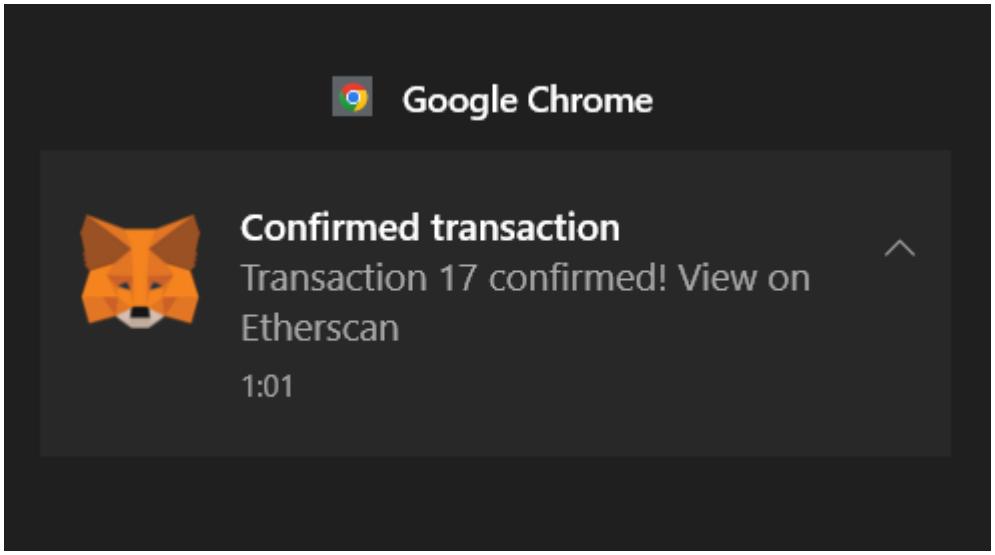

Рисунок 4.12 – Здійснена успішна транзакція

Задля тестування смарт-контракту ArcaneToken слід відкрити сторінку за адресою 0x31c43c8Adee697c9608FF77dE8fd408B6Ec52945 на досліднику Binance Testnet (рисунок 4.13):

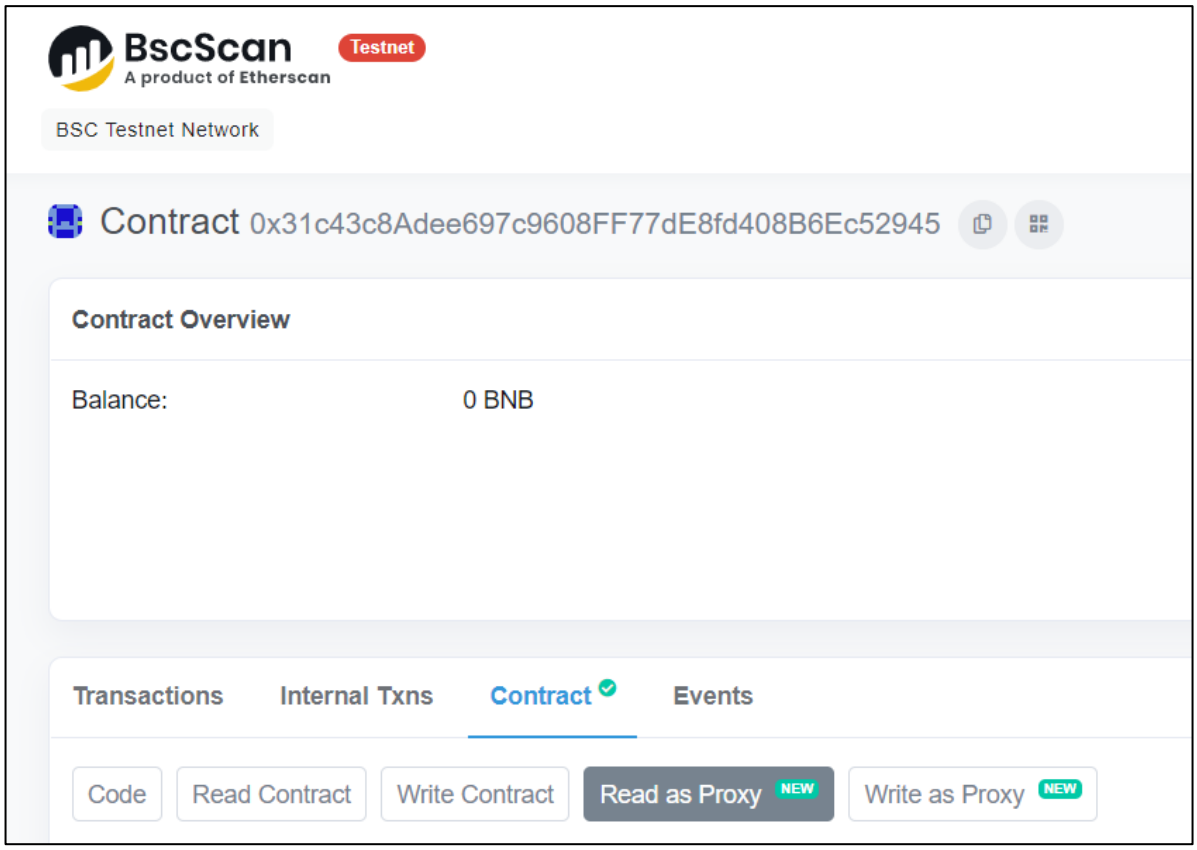

Рисунок 4.13 – Сторінка тестового токена

BscScan надає можливість переглянути такі дані: баланс смарт-контракту; овнера розгорнутого контракту та всієї його активності; стан відправлених транзанкцій на адресу; стан внутрішніх операцій; транзакції, пов’язані з криптоактивами; лістинг коду смарт-контракту, версію компілятора, налаштування, задані при розгортанні контракту; ABI, байт-код, закодовані аргументи конструктора тощо; згенеровані події; дані смарт-контракту, записані у публічних змінних, або результат повернення view-методів.

Після успішної транзакції в додатку Arcane користувач може переглянути список транзакцій на адресі смарт-контракту (рисунок 4.14):

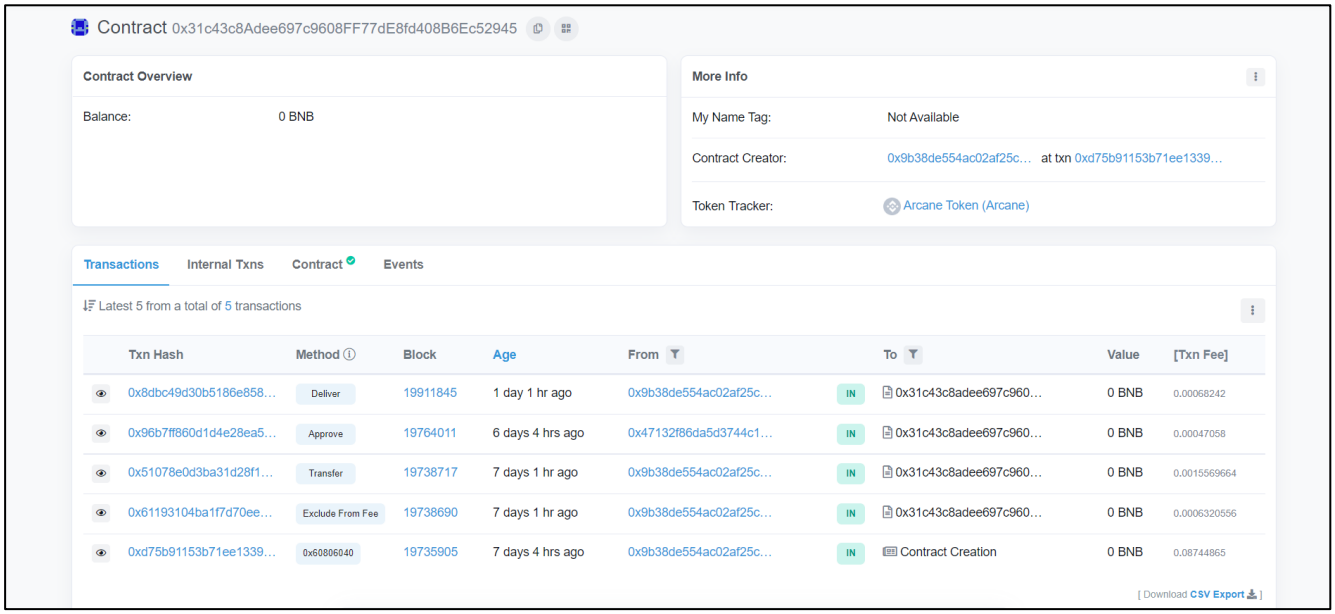

Рисунок 4.14 – Список транзакцій тестового пула після схвалення токенів

Як було вищезгадано, користувач має можливість дізнатись публічну інформацію безпосередньо на тестовій мережі BscScan (рисунок 4.15):

Рисунок 4.15 – Баланс Arcane-токенів овнера на BscScan Testnet

Після здійснення користувачем кількох транзакцій його баланс змінюється. Перевірити це можна в розширенні MetaMask за адресою обраного гаманця на вкладці assets (рисунок 4.16):

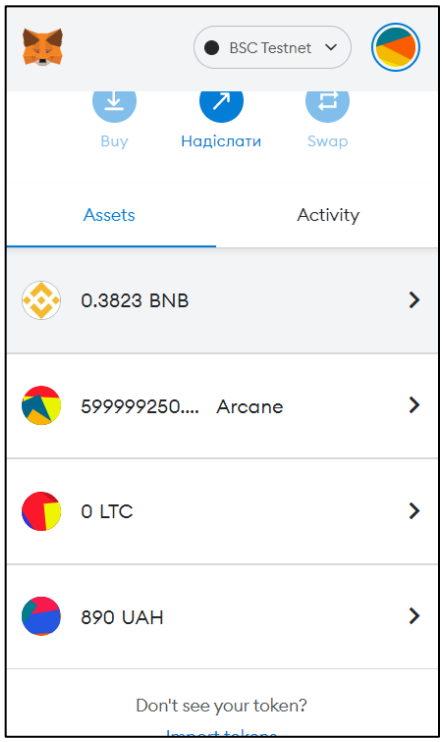

Рисунок 4.16 – Змінені assets на обраній адресі гаманця

В будь-якому випадку при надсиланні транзакції користувач повинен заплатити комісію в MetaMask, після чого транзакція зможе бути успішно опрацьованою (рисунок 4.17):

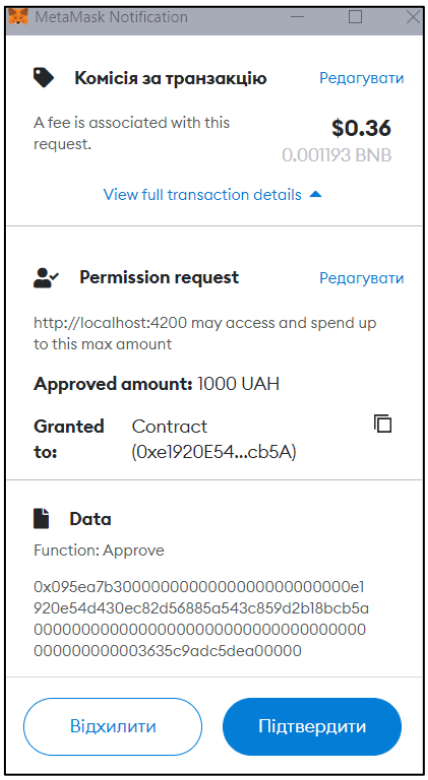

Рисунок 4.17 – Оплата комісії за здійснення транзакції

У випадку спроби виконання транзакції користувачем, що не відповідає умовам контракту, надійде сповіщення щодо некоректності розрахунку необхідного газу для транзакції та можливу помилку умов договору. В результаті надсилання такої транзакції стан контракту не зміниться, проте заплатити доведеться більшу суму комісії з метою покриття потужностей, що були задіяні валідаторами блоку при опрацюванні умов контракту, що зображено на рисунку 4.18:

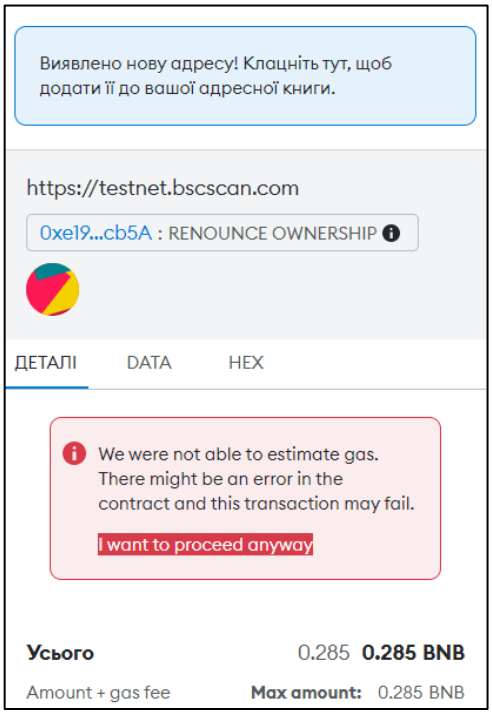

Рисунок 4.18 – Попередження про невдале здійснення транзакції

За бажанням кожен користувач, навіть той, що не здійснив з'єднання гаманця з додатком, має можливість підтримати створений проєкт та здійснити інвестицію в нього, а також в перспективі стати повноцінним партнером Arcane-команди та потрапити в так званий whitelist – список користувачів, яким на контракті токена нараховуються нагороди та всі транзакції вони здійснюють без оплати додаткових комісій, зафіксованих овнером. Модальне вікно для надсилання коштів зображено на рисунку 4.19:

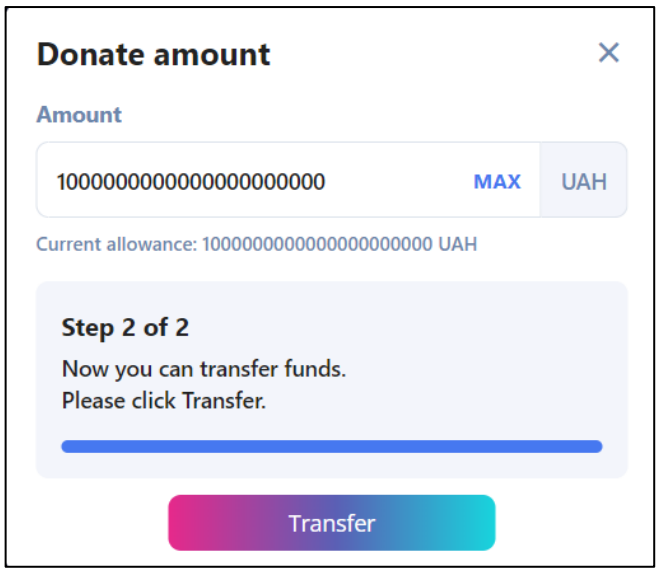

Рисунок 4.19 – Можливість інвестувати в створений проєкт

5 ЕКОНОМІЧНА ЧАСТИНА

Блокчейн-платформи, побудовані на основі децентралізованих фінансових сервісів, почали створювати близько 5 років тому. В період 2021-2022 років DeFi набуває все більшої популярності, виправдовуючи поставлені цілі. Порівняно з іншими програмними застосунками, створеними за іншими технологіями, блокчейн-додатки усувають більшість можливих вразливостей як при конструюванні, так і під час використання. Попри популярність створення бізнес-ідей та втілення їх через застосунок в світі існує досить мало розробників, які, незважаючи на складність та іноваційність, займаються аналізом міжнародного ринку криптовалют, визначенням практичніших методів конструювання, написанням, перевіркою та розгортанням смарт-контрактів на мережах.

Arcane – унікальна платформа, за допомогою якої користувач має можливість створювати будь-який вестінг на вибір і отримувати лінійний дохід, залежно від часових рамок, за рахунок використання Arcane-токенів, при цьому накопичуючи на балансі певну суму нагород. З часом є можливість потрапити в спеціальний список типу whitelist і здійснювати фінансові операції без додаткових комісій.

Аналізуючи поточний міжнародний ринок криптовалютних проєктів, знайдено кілька схожих за бізнес-логікою платформ, зокрема DAO Maker та Aave-платформа. В обох додатках забезпечено можливість розвивати і підтримувати подальші проєкти безпечно, відповідно до протоколу за рахунок продуманого вестінга. Проте жодна компанія не розгортала контракти на мережі Binance, відповідно в них не є основною нативною криптовалютою BNB. Це одна з найсуттєвіших відмінностей між Arcane-застосунком і його аналогами.

Враховуючи всі вищенаведені фактори, цілком зрозуміло причину високої вартості типових програмних продуктів на ринку. Ціновий діапазон найпростіших аналогічних додатків без жодних розширених можливостей на міжнародному ринку регулюється від $3,000 та вище. Враховуючи інтеграцію складніших механізмів блокчейну, а також аудит безпеки коду смарт-контрактів, ціна за продукт зростатиме ще більше.

5.1 Розрахунок трудомісткості програмного продукту

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки окремих етапів об'єму функцій програмних засобів в залежності від їх типів, а також кваліфікації робітників та запланованих строків, що диктують умови ринку.

Загальну трудомісткість можна визначити за формулою:

$$\mathrm{T}_{\text{заг}} = N_{\text{час}} \cdot k_{\text{скл}} \cdot k_{\text{м}} \cdot k_{\text{станд}} \cdot k_{\text{станд.ПП}} \qquad (5.1)$$

де:

$\mathrm{T}_{\text{заг}}$ – загальна трудомісткість, людино-дні;

$N_{\text{час}}$ – норма часу, людино-дні;

$k_{\text{скл}}$ – коефіцієнт складності контролю вхідної та вихідної інформації;

$k_{\text{м}}$ – коефіцієнт використання мови програмування певного рівня складності;

$k_{\text{станд}}$ – коефіцієнт використання стандартних програм;

$k_{\text{станд.ПП}}$ – коефіцієнт розробки стандартного програмного продукту (ПП).

Розробляючи ПП, використовуються стандартні модулі і пакети прикладних програм чи типові програми, тому норму часу коригують за допомогою коефіцієнта $k_{\text{станд}}$ = 0,6-0,8.

За формулою 5.1 виконано обчислення:

$$\mathrm{T}_{\text{заг}} = 25 \cdot 1{,}08 \cdot 1 \cdot 0{,}8 \cdot 1{,}5 = 33 \text{ (людино/дні)}$$

5.2 Розрахунок собівартості програмного продукту

Основним показником, що характеризує рівень витрат на виробництво продукції є собівартість продукції.

Собівартість продукції як показник використовується для контролю за використанням ресурсів виробництва, визначення економічної ефективності організаційно-технічних заходів, встановлення цін на продукцію.

Фактична або повна собівартість ПП визначається в процесі проведення калькуляції собівартості та є сумою виробничої собівартості, адміністративних витрат та витрат на збут.

5.2.1 Обчислення витрат на заробітну плату

До заробітної плати розробників ПП ($С_{ЗПр}$) належать витрати на виплату основної та додаткової зарплати виконавців, обчислені згідно із системою оплати праці, прийнятими в організації, включаючи будь-які види матеріальних та грошових доплат.

Заробітна плата розробників ПП визначається за формулою:

$$C_{ЗПр} = C_{ЗПосн} + C_{ЗПдод} \tag{5.2}$$

Основна заробітна плата – це винагорода за виконану роботу відповідно до встановлених норм праці. Вона встановлюється у вигляді тарифних ставок (окладів) і відрядних розцінок для робітників та посадових окладів для службовців. Заробітна плата розробників ПП визначається за формулою:

$$C_{ЗПосн} = C_{ЗПден} \cdot T_{заг} \tag{5.3}$$

де:

$С_{ЗПден}$ – денна зарплата програміста, грн;

$Т_{заг}$ – загальна трудомісткість розробки ПП, людино-дні.

Денну заробітну плату визначають, виходячи з місячних окладів:

$$C_{ЗПден} = \frac{Сок \cdot 12}{Фрч} \tag{5.4}$$

де:

$С_{ок}$ – місячний оклад розробника ПП, грн;

$Ф_{рч}$ – річний фонд робочого часу, днів.

Трудовим законодавством встановлено на 2022 рік: $Фрч = 249$ днів. Місячний оклад розробника ПП встановлено у розмірі 25000 грн.

$$С_{ЗПден} = \frac{25000{,}00 \cdot 12}{249} = 1204{,}82 \text{ (грн)}$$

Отож, основна заробітна плата складатиме:

$$С_{ЗПосн} = 1204{,}82 \cdot 27 = 32530{,}14 \text{ (грн)}$$

Додаткова заробітна плата (премії, одноразові заохочення тощо) розраховується згідно з нормативом, який установлює підприємство і який складає

10 – 40 % від основної зарплати. Витрати на додаткову заробітну плату визначаються за формулою:

$$С_{ЗПдод} = С_{ЗПосн} \cdot k_{ЗПдод} \quad (5.5)$$

де:

$k_{ЗПдод}$ – нормативний коефіцієнт додаткової заробітної плати (становить 30%);

$С_{ЗПосн}$ – витрати на основну заробітну плату, грн.

$$С_{ЗПдод} = 32530{,}14 \cdot 30\ \% = 9759{,}04\ (грн)$$

Заробітна плата розробника ПП за формулою 5.2 складе:

$$С_{ЗПр} = 32530{,}14 + 9759{,}04 = 42289{,}18\ (грн)$$

5.2.2 Обчислення єдиного соціального внеску

Єдиний соціальний внесок – це обов'язкові відрахування на загальнодержавне соціальне страхування. ЄСВ сплачується підприємцем за себе та за кожного найманого робітника. Він є внеском у загальнодержавну систему соціального страхування з метою захисту у випадках, передбачених законодавством, прав застрахованих осіб на отримання страхових виплат.

Згідно з нормами частини п'ятої ст. 8 Закону України про ЄСВ єдиний внесок для всіх платників єдиного внеску (крім пільгових категорій) встановлено у розмірі 22 % до визначеної ст. 7 цього Закону бази нарахування єдиного внеску.

Таким чином, витрати на єдиний соціальний внесок становитимуть:

$$42289{,}18\ \cdot 22\ \% = 9303{,}62\ (грн)$$

5.2.3 Обчислення експлуатаційних витрат

Окрім витрат пов'язаних з нарахуванням заробітної плати, при розробці програмного продукту виникають експлуатаційні витрати. Ці витрати пов'язані з використанням технічних засобів, електроенергії тощо. Експлуатаційні витрати вираховуються згідно з формулами по даних, які представлено у таблиці 5.1.

Таблиця 5.1 – Дані для розрахунку експлуатаційних витрат, пов’язаних з розробкою програмного продукту

Найменування показника	Одиниця виміру	Значення
Вартість комп’ютера	грн.	28000,00
Тариф на 1 кВт електроенергії	грн.	1,44
Потужність комп’ютера	кВт/год.	0,063
Тариф Інтернету	грн./місяць	190,00

Крім того, при розробці програмного продукту використовуються розхідні матеріали, витрати яких складають 3% від заробітної плати. В даному випадку, величина витрат складатиме:

$$42289{,}18 \cdot 3\% = 1268{,}68 \text{ (грн)}$$

До уваги ще слід взяти зношування комп’ютера при експлуатації. Процес відшкодування зношування здійснюється шляхом амортизації.

Амортизація – процес поступового перенесення вартості основних засобів на продукт, що виготовляється з їх допомогою. Для заміщення зношеної частини основних засобів виробництва підприємства роблять амортизаційні відрахування, тобто відрахування певних грошових сум відповідно до розмірів фізичного і морального зносу засобів виробництва.

За прямолінійним методом річна сума амортизації визначається діленням вартості, яка амортизується, на строк корисного використання об’єкта основних засобів. Вартість об’єкта, що амортизується, рівномірно списується протягом строку його служби. При цьому річна норма та річна сума амортизації залишаються постійними на весь строк корисної служби об’єкта.

Мінімальний строк корисного використання визначений в Податковому кодексі України, становить від 2 до 20 років в залежності від групи. Комп’ютер відносить до 4-ї групи основних засобів, тобто мінімальний строк корисного використання становить 10 років.

За прямолінійним методом, річна амортизація становить:

$$\frac{28000{,}00}{10} = 2800{,}00\ (\text{грн})$$

Вартість амортизації на розробку даного програмного продукту визначається за формулою:

$$\frac{\text{Річна амортизація}}{\text{Кількість днів у році}} \cdot \text{Загальна трудомісткість} \qquad (5.6)$$

Згідно вищесказаної формули, амортизація на розробку даного програмного продукту становить:

$$\frac{2800{,}00}{365} \cdot 33 = 253{,}15\ (\text{грн})$$

5.2.4 Обчислення витрат на електроенергію

Розрахунок вартості енергоносіїв здійснюється на основі визначеної фактичної кількості використаних енергоносіїв та тарифів. Згідно з тарифами, станом на 1 травня оплата за електроенергію становить: до 250 кВ – 1,44 грн., більше 250 кВт – 1,68 грн. за один кВт.

Витрати на електроенергію при розробці програми розраховуються за формулою:

$$\mathrm{B_e} = \mathrm{P} \cdot \mathrm{T_{заг}} \cdot N_{\text{год}} \cdot \mathrm{T_e} \qquad (5.7)$$

де:

$\mathrm{B_e}$ – витрати на електроенергію;

P – потужність ЕОМ (кВт/год);

$\mathrm{T_{заг}}$ – загальна трудомісткість розробки ПП;

$N_{\text{год}}$ – тривалість робочого дня;

$\mathrm{T_e}$ – тариф електроенергії.

Відповідно до вищесказаної формули, витрати на електроенергію становлять:

$$0{,}063 \cdot 33 \cdot 8 \cdot 1{,}44 = 23{,}95\ (\text{грн})$$

5.2.5 Розрахунок інших виробничих витрат

Інші виробничі витрати становлять 30% від зарплати розробника, а також витрати на використання Інтернету. Використання Інтернету складає 60% всього

часу відведеного на роботу за комп'ютером.

Час роботи з Інтернетом $= \mathrm{T_{заг}} \cdot$ Кількість робочих годин $\cdot 60\ \%$

де $\mathrm{T_{заг}}$ – загальна трудомісткість роботи.

Отже, час роботи з Інтернетом $= 33 \cdot 8 \cdot 60\ \% = 158{,}4$ (год)

Витрати за Інтернет розраховуються за формулою:

$$\mathrm{B_i} = \frac{\mathrm{T_i}}{N_{\text{днів}} \cdot N_{\text{год}}} \cdot \mathrm{T} \qquad (5.8)$$

де:

$\mathrm{B_i}$ – витрати на Інтернет;

$\mathrm{T_i}$ – тариф Інтернету;

$N_{\text{днів}}$ – кількість робочих днів у місяці;

$N_{\text{год}}$ – тривалість робочого дня;

Т – час роботи з Інтернетом.

За вищесказаною формулою, витрати на Інтернет становитимуть:

$$\frac{190{,}00}{21 \cdot 8} \cdot 158{,}4 = 179{,}14(\text{грн})$$

Отже, інші виробничі витрати становлять: $42289{,}18(\text{грн}) \cdot 30\ \% + 179{,}14(\text{грн}) = 12865{,}89$ (грн)

Окрім вказаних поточних витрат на розробку ПП, собівартість розробника реалізації ПП передбачає визначення:

– адміністративні витрати (організаційні витрати, витрати на службові відрядження, страхування, амортизацію, опалення, освітлення, водопостачання, охорону; винагорода за професійні послуги: юридичні, аудиторські; витрати на зв'язок; витрати за послуги банку);

– витрати на збут (на рекламу та дослідження ринку: маркетинг; витрати на гарантійний ремонт і гарантійне сервісне обслуговування; комісійні витрати; витрати, пов'язані з безпосереднім постачанням: страхування, амортизація, охорона).

Адміністративні витрати належать до собівартості ПП пропорційно основній заробітній платі складають 15 – 30% від основної заробітної плати.

Витрати на збут за цією статтею витрати визначаються у відсотках (3 – 5%)

від виробничої собівартості.

Приймемо розмір адміністративних витрат 10%, а витрат на збут 5%.

Адміністративні витрати:

$$32530{,}14 \cdot 10\% = 3253{,}01 \text{ (грн)}$$

Витрати на збут:

$$48918{,}50 \cdot 5\% = 2445{,}93 \text{(грн)}$$

5.2.6 Розрахунок собівартості програмного продукту

Розрахунок витрат на розробку програмного продукту наведено в таблиці 5.2

Таблиця 5.2 – Витрати на розроблення веб-застосунку Arcane

Статті калькуляції	Вартість, грн.
Витрати на заробітну плату	42289,18
Єдиний соціальний внесок	9303,62
Витрати на розхідні матеріали	1268,68
Амортизаційні відрахування	327,67
Витрати на електроенергію	23,95
Інші виробничі витрати	12865,89
Повна собівартість	66078,99
Адміністративні витрати	3253,01
Витрати на збут	2445,93
Повна (фактична) собівартість програмного продукту	71777,93

5.3 Розрахунок ціни програмного продукту

Ціна ПП визначається як сума повної собівартості продукту і величини прогнозованого прибутку.

Для прибутковості ПП прийнято величину прибутку обчислювати за рівнем рентабельності в розмірі 10-30 %.

Прибуток (П) розраховується згідно з рівнем рентабельності (Р), як відсоток від повної собівартості ПП за формулою:

$$П = С_{повн} \cdot Р \quad (5.9)$$

Тоді ціну виробу можна визначити за формулою:

$$Ц = С_{повн} + П \quad (5.10)$$

Ціна реалізації продукції за умови рівня рентабельності 30% розраховується:

Величина прибутку становитиме:

$$П = 71777{,}93 \cdot 30\ \% = 21533{,}38\ (грн)$$

Тоді ціна на ПП складе:

$$Ц = 71777{,}93 + 21533{,}38 = 93311{,}31\ (грн)$$

5.4 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Ер) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_р = \frac{П}{С_В} \quad (5.11)$$

де:

П – прибуток;

СВ – собівартість.

Плановий прибуток (Ппл) знаходимо за формулою:

$$П_{ПЛ} = Ц - С_В \quad (5.12)$$

Розраховуємо плановий прибуток: Ппл = 93311,31 – 71777,93 = 21533,38 (грн)

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П_{ПЛ}}{С_В} \quad (5.13)$$

Тоді, Ер = 21533,38 / 71777,93 = 0,3.

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (Тр):

Термін окупності дорівнює:

Тр = 1 / 0,3 = 3 місяці

В результаті виконання всіх вищесказаних розрахунків встановлено, що продукт є конкурентоспроможним, використана технологія розробки програмного продукту відповідає оптимальному рівню витрат, додаток є прибутковим та є економічно доцільним. За заданим рівнем рентабельності продукт дає достатній рівень прибутку.

Проаналізувавши всі отримані результати, можна дійти висновку, що додаток є економічно доцільним та конкурентоспроможним для використанню на підприємстві чи для започаткування власної справи в компанії Applicature.

6 ОХОРОНА ПРАЦІ

6.1 Вимоги до користувачів при експлуатації ПК

Положення для користувачів персональних комп’ютерів з охорони праці:

а) виконувати умови інструкції з експлуатації ПК;

б) при експлуатації ПК необхідно пам'ятати, що первинні мережі електроспоживання під час роботи знаходяться під напругою, яка є небезпечною для життя людини, тому необхідно користуватися справними розетками, відгалужувальними та з'єднувальними коробками, вимикачами та іншими електроприладами;

в) до роботи з ПК допускаються працівники, з якими проведений вступний інструктаж та первинний інструктаж (на робочому місці) з питань охорони праці, техніки безпеки, пожежної безпеки та зроблений запис про їх проведення у спеціальному журналі інструктажів;

г) працівники при роботі з ПК повинні дотримуватися вимог техніки безпеки, пожежної безпеки;

д) при виявленні в обладнанні ПК ознак несправності (іскріння, пробоїв, підвищення температури, запаху гару, ознак горіння) необхідно негайно припинити роботи, відключити усе обладнання від електромережі і терміново повідомити про це відповідних посадових осіб, спеціалістів;

е) вміти діяти в разі ураження інших працівників електричним струмом або виникнення пожежі;

ж) знати місця розташування первинних засобів пожежогасіння, план евакуації працівників, матеріальних цінностей з приміщення в разі виникнення пожежі;

з) не допускати ушкодження чи модифікування шнура живлення. Заборонено ставити важкі речі на шнур, тягнути чи надмірно перегинати його, скручувати та перев’язувати шнур вузлом;

и) перед початком виконання роботи і після її завершення обов’язково перевіряти стан апаратури, справність електропроводки, з’єднувальних шнурів,

штепсельних вилок, розеток, заземлення захисного екрана;

к) про всі виявлені несправності інформувати керівника робіт і не братися до роботи, доки їх не буде усунено;

л) при роботі з ПК заборонено самостійно розбирати та ремонтувати системний блок (корпус ноутбука), монітор, клавіатуру, комп'ютерну мишу тощо, встромляти сторонні предмети до вентиляційних отворів ПК, ноутбука або монітора, ставити на системний блок ПК та периферійні пристрої металеві предмети та ємкості з водою;

м) перед початком роботи самостійно відрегулювати яскравість освітлення, контрастність монітора [26].

6.2 Аналіз шкідливих впливів на користувачів комп'ютерної техніки

Розробники програмного забезпечення переважно задіяні на роботах, пов'язаних з періодичною або постійною роботою за комп'ютером, піддаються впливу факторів виробничої небезпеки. До основних шкідливих факторів при роботі з комп'ютером відносять: тривале сидяче положення, електромагнітне випромінювання, навантаження на зір, перевантаження кистьових суглобів, можливість захворювань органів дихання, алергії та ін. Тривале сидяче положення приводить до напруги м'язів шиї, голови, рук і плечей, остеохондрозу, у дітей – ще й до сколіозу. Окрім цього, воно ще приводить до застою крові в тазових органах і, як наслідок, до простатиту й геморою. Не секрет, що малорухливий спосіб життя призводить до ожиріння. Остеохондроз виникає при порушенні міжхребцевих дисків, яке призводить до випинання в яку або сторону (грижі міжхребцевого диска). Грижа може негативно впливати на спинний мозок і нервові відростки. Наслідки можуть бути найрізноманітнішими, від болів в спині і кінцівках, до паралічу кінцівок і смерті. Одна з поширених причин остеохондрозу - дистрофія м'язів спини. Людина, що проводить в основному сидячий спосіб життя, має велику ймовірність захворіти остеохондрозом [27].

Останнім часом частіше з'являються повідомлення про виникнення комп'ютерної залежності. Дійсно, тривала робота за комп'ютером може викликати

психічні розлади. Вона нерідко поглинає всю увагу працівника, через що він часто нехтує нормальним харчуванням. Неправильне харчування призводить не тільки до порушень роботи органів травного тракту, але і до виникнення мінеральної і вітамінної недостатності. Відомо, що нестача вітамінів і мінералів негативно позначається на процесі обміну речовин в організмі, що призводить до зниження інтелектуальних здібностей людини. Таким чином, утворюється своєрідне «порочне коло», в якому тривала робота за комп'ютером є пусковим моментом, який визначає всі наступні порушення.

Гіподинамія, стрес, шкідливі звички і неправильне харчування є головними причинами серцево-судинних захворювань і діабету. Таким чином, людина, що тривалий час працює за комп'ютером, має велику ймовірність виникнення серцево-судинних захворювань, різних захворювань очей, рухового апарату, органів шлунково-кишкового тракту, психічних розладів.

Дисплейна хвороба, характеризується порушенням акомодації очей через тривале перенапруження війкового тіла. Війкове тіло розташоване відразу під веселковою оболонкою ока і складається з безлічі м'язових волокон. Війкове тіло являє собою своєрідне м'язове кільце, усередині якого кріпиться кришталик. Скорочення або розслаблення м'язів війкового тіла приводить до зміни кривизни кришталика і, отже, змінює його заломлюючу здатність. У нормі робота війчастих тіл обох очей підтримує концентрування світлового пучка на обмежену ділянку сітківки. При хронічному перенапруженні війкового тіла воно втрачає здатність скорочуватися а, отже, втрачається здатність очей до акомодації (сприйняття об'єктів на різних відстанях).

Синдром сухого ока – збірна назва захворювання викликаного порушенням зволоження передньої поверхні ока (рогівки) слізної рідиною. У нормі людина здійснює більше 20 моргальних рухів в секунду. У результаті цього передня поверхня ока постійно зволожується і очищується слізної рідиною. Під час роботи за комп'ютером частота моргання зменшується щонайменше в три рази. При цьому поверхня рогівки «висихає». Синдром сухого ока розвивається через деякий час роботи за комп'ютером і проявляється печінням в очах, почервонінням,

кон'юнктивітами, появою судинної сітки на бічних поверхнях очей. Якщо при виникненні цих ознак робота за комп'ютером припиняється, то симптоми регресують. Однак під час тривалої роботи за комп'ютером вищевказані симптоми стають більш стійкими і не зникають після припинення роботи на комп'ютері. Пояснюється це приєднанням інфекції і порушенням трофіки оболонок ока, викликані недостатнім зволоженням очей слізної рідиною.

Також тривала робота за комп'ютером може збільшити ризик таких очних захворювань як міопія (короткозорість), далекозорість, глаукома.

Тривала робота за комп'ютером може стати причиною серйозних нервово-м'язових розладів. Особливо чутливими ділянками тіла є пальці, кисті рук та передпліччя. Руки виконують основну частину механічної роботи при роботі за комп'ютером, при цьому важлива не амплітуда фізичного навантаження (вона, як правило, досить низька), а час роботи. Як відомо подушечки пальців є найбільш чутливими ділянками людського тіла. На цьому рівні сконцентрована велика кількість чутливих нервових закінчень (завдяки цьому пальці виконують функцію дотику). При тривалій роботі за комп'ютером (на клавіатурі) нервові закінчення пальців піддаються постійному роздратування. З часом це призводить до виснаження нервових шляхів здійснюють зв'язок пальців з корою головного мозку. В результаті виникають порушення координації рухів пальців і судоми кисті та передпліччя.

Робота за комп'ютером − це інтелектуальна праця, через це основна частина навантаження припадає на нервову систему, а саме на головний мозок. Часто тривала робота за комп'ютером може бути причиною головних болів. Одним з чинників, що провокує появу головних болів є хронічне перенапруження, важливе значення має і постійна напруга черепних м'язів і м'язів обличчя. Розлади уваги і неможливість концентруватися є наслідком хронічної перевтоми. Іноді через тривалу роботу за комп'ютером може виникнути шум у вухах, запаморочення, нудота. При виникненні цих симптомів потрібно звернутися за порадою до лікаря і тимчасово перервати роботу за комп'ютером.

6.3 Основні вимоги до робочого місця з використанням ПК

На робочому місці повинні бути передбачені заходи захисту від можливого впливу небезпечних і шкідливих факторів виробництва. Рівні цих факторів не повинні перевищувати граничних значень, обговорених правовими, технічними й санітарно-технічними нормами. Ці нормативні документи зобов'язують до створення на робочому місці умов праці, при яких вплив небезпечних і шкідливих факторів усунуто або він перебуває в припустимих межах. Правильно спроектоване й виконане виробниче освітлення покращує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, підвищує безпеку праці й знижує травматизм.

Недостатність освітлення призводить до напруги зору, послаблює увагу, призводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає осліплення, роздратування й різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть привести до нещасного випадку або профзахворювань, тому важливий правильний розрахунок освітленості.

Існує три види освітлення – природне, штучне та сполучене (природне й штучне разом) [28].

Природне освітлення – освітлення приміщень денним світлом, що проникає через світлові прорізи в зовнішніх конструкціях, що обгороджують, приміщень. Природне освітлення характеризується тим, що міняється в широких межах залежно від часу дня, пори року, характеру області й ряду інших факторів.

Штучне освітлення застосовується при роботі в темний час доби й удень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день).

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, у свою чергу, може бути загальним або комбінованим. Загальне – освітлення, при якому світильники розміщаються у верхній зоні приміщення рівномірно або стосовно до розташування встаткування. Комбіноване - освітлення, при якому до загального додається місцеве освітлення.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, що випливають: при виконанні зорових робіт високої точності загальна освітленість повинна становити 300 лк, а комбінована – 750 лк; аналогічні вимоги при виконанні робіт середньої точності – 200 й 300 лк відповідно.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури й зниженню відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 установлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються залежно від пори року, характеру трудового процесу й характеру виробничого приміщення (таблиця 6.1).

Таблиця 6.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22-24 С
	Відносна вологість	40-60%
	Швидкість руху повітря	до 0,1м/с
Теплий	Температура повітря в приміщенні	23-25 С
	Відносна вологість	40-60%
	Швидкість руху повітря	0,1-0,2м/с

Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, наведені в таблиці 6.2.

Таблиця 6.2 – Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери

Характеристика приміщення	Об'ємна витрата подаваного в приміщення свіжого повітря, $м^3$/на одну людину в годину
обсяг до 20$м^3$ на людину	не менш 30

Кінець таблиці 6.2

20-40м^3 на людину	не менш 20
більше 40 м^3 на людину	природна вентиляція

Для забезпечення комфортних умов використаються як організаційні методи (раціональна організація проведення робіт залежно від пори року й доби, чергування праці й відпочинку), так і технічні засоби (вентиляція, кондиціонування повітря, опалювальна система).

Персональні комп'ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію [29].

Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту. У приміщенні, де одночасно експлуатуються понад п'ять персональних комп'ютерів і периферійних пристроїв, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Персональні комп'ютери і периферійні пристрої повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. У штепсельних з'єднаннях та

електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників.

Порядок роз'єднання при відключенні має бути зворотним. Не допускається підключати персональні комп'ютери та периферійні пристрої до звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв. Електромережу штепсельних розеток для живлення персональних комп'ютерів і периферійних пристроїв при розташуванні їх уздовж стін приміщення прокладають по підлозі поруч зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах, а також у пластикових коробах і пластмасових рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розміщенні в приміщенні до п'яти персональних комп'ютерів і периферійних пристроїв допускається прокладання трипровідникового захищеного проводу або кабелю в оболонці з негорючого чи важкогорючого матеріалу по периметру приміщення без металевих труб та гнучких металевих рукавів. Не допускається в одній трубі прокладати ланцюги до 42В та вище 42В. При організації робочих місць операторів електромережу штепсельних розеток для живлення персональних комп'ютерів, периферійних пристроїв і у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку.

6.4 Розрахунок штучного освітлення приміщення

Для створення сприятливих умов зорової роботи без швидкої втомлюваності очей, виникнення професійних захворювань, нещасних випадків, які би сприяли підвищенню продуктивності праці та якості продукції, виробниче освітлення повинно відповідати наступним вимогам:

1) створювати на робочій поверхні освітленість, що відповідає характеру

зорової роботи і не є нижчою за встановлені норми;

2) забезпечити достатню рівномірність та постійність рівня освітленості у виробничих приміщеннях, щоб уникнути частої переадаптації органів зору;

3) не створювати засліплювальної дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

4) не створювати на робочій поверхні різких та глибоких тіней (особливо рухомих);

5) повинен бути достатній для розрізнення деталей контраст поверхонь, що освітлюються;

6) не створювати небезпечних та шкідливих виробничих чинників (шум, теплові випромінювання, небезпека ураження струмом, пожеж та вибухонебезпеки світильників);

7) повинно бути надійним, простим в експлуатації та економічним.

При виконанні розрахунку освітлення перш за все необхідно знайти індекс приміщення:

$$I = \frac{A \cdot B}{H_p (A + B)}, \tag{6.1}$$

де:

A і B – це відповідно довжина і ширина приміщення, м;

H_p – висота, м.

Маючи значення I в таблиці, що подана в додатку В знаходимо значення η, що буде залежати також від коефіцієнтів відбиття та типу світильника.

Кількість світильників із умови забезпечення рівномірності освітлення розраховують за формулою

$$N = \frac{A \cdot B}{L^2}, \tag{6.2}$$

де L – рекомендована відстань між світильниками і визначається за формулою (6.3):

$$L = 0{,}6 \cdot H_{p,} \tag{6.3}$$

де H_p – висота приміщення, м.

Визначаємо необхідний світловий потік ламп в кожному ряду:

$$\Phi = \frac{E_H \cdot S \cdot Z \cdot k}{N_p \cdot \eta}, \quad (6.4)$$

де:

Φ – необхідний світловий потік, лм;

$E_н$ – значення освітлення, яке треба реалізувати, лк;

S – площа приміщення, м^2;

Z – коефіцієнт нерівномірності освітлення;

k – коефіцієнт запасу;

N_p – кількість світильників в ряду;

η – коефіцієнт використання.

Таблиця 6.3 – Технічні дані люмінесцентних ламп

Тип лампи	Потужність, Вт	Розрахунковий світловий потік, лм
ЛДЦ 40-4	40	1995
ЛД 40-4	40	2225
ЛХБ 40-4	40	2470
ЛТБ 40-4	40	2450
ЛБ 40-4	40	3000
ЛХБЦ 40-4	40	2000
ЛДЦ 80-4	80	3380
ЛХБ 80-4	80	4220
ЛТБ 80-4	80	4300
ЛБ 80-4	80	4960

Для визначення необхідної кількості світильників в ряду користуємось формулою (6.5).

$$N = \frac{\Phi_p}{n \cdot \Phi_л}, \quad (6.5)$$

де:

Φ_p – це світловий потік в ряді, лм;

n – кількість лампочок в світильнику;

$Ф_л$ – світловий потік лампи, лм.

Технічні дані люмінесцентних ламп подані в таблиці 6.3

В цеху розміром 16×25×4 м потрібно реалізувати освітленість 350 лк ($E_н$). Коефіцієнт відбивання стелі $\rho_{стелі}$ = 70 %, коефіцієнт відбивання стін $\rho_{стін}$ = 50 %. Для освітлення використовують люмінесцентні лампи типу ЛТБ 40-4 в світильниках ПВЛП. Приймаємо коефіцієнт запасу k=1,6 і коефіцієнт нерівномірності освітлення Z=1,1.

Таблиця 6.4 – Значення коефіцієнта використання світлового потоку світильників з люмінесцентними лампами

Коефіцієнт	Тип світильника														
відбиття	ПВЛМ-Д			ПВЛМ-ДО			ПВЛМ-ДР			ЛДОР			ПВЛП		
$\rho_{стелі,\%}$	70	50	30	70	50	30	70	50	30	70	50	30	70	50	30
$\rho_{стін,\%}$	50	30	10	50	30	10	50	30	10	50	30	10	50	30	10
$\rho_{п,\%}$	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Індекс, I	Коефіцієнт використання, η														
0,5	0,27	0,21	0,18	0,28	0,20	0,16	0,24	0,20	0,17	0,25	0,19	0,14	0,18	0,13	0,11
0,6	0,32	0,25	0,22	0,32	0,24	0,20	0,31	0,25	0,21	0,29	0,22	0,18	0,23	0,17	0,14
0,7	0,36	0,30	0,26	0,36	0,29	0,24	0,35	0,29	0,26	0,33	0,26	0,22	0,27	0,20	0,16
0,8	0,39	0,33	0,29	0,40	0,32	0,27	0,38	0,32	0,28	0,36	0,30	0,25	0,29	0,23	0,19
0,9	0,42	0,37	0,32	0,43	0,36	0,30	0,41	0,36	0,32	0,40	0,33	0,28	0,32	0,26	0,21
1,0	0,45	0,40	0,35	0,46	0,39	0,33	0,44	0,39	0,35	0,43	0,36	0,30	0,34	0,28	0,23
1,1	0,48	0,42	0,38	0,49	0,41	0,35	0,46	0,41	0,37	0,45	0,38	0,32	0,36	0,30	0,25
1,2	0,50	0,45	0,40	0,52	0,44	0,38	0,48	0,43	0,39	0,47	0,40	0,35	0,38	0,32	0,27
1,25	0,54	0,49	0,45	0,56	0,48	0,42	0,52	0,48	0,44	0,51	0,44	0,38	0,42	0,36	0,30
1,5	0,57	0,52	0,48	0,59	0,52	0,46	0,55	0,51	0,47	0,54	0,47	0,42	0,44	0,38	0,33
1,75	0,59	0,55	0,51	0,61	0,54	0,48	0,57	0,53	0,49	0,56	0,49	0,44	0,46	0,40	0,35
2,0	0,62	0,57	0,53	0,64	0,56	0,50	0,59	0,55	0,51	0,58	0,51	0,46	0,48	0,42	0,37
2,25	0,63	0,58	0,55	0,66	0,58	0,52	0,60	0,56	0,53	0,60	0,53	0,48	0,50	0,43	0,39
2,5	0,65	0,61	0,58	0,68	0,60	0,55	0,62	0,58	0,55	0,62	0,55	0,50	0,52	0,45	0,41
3,0	0,67	0,62	0,60	0,69	0,62	0,57	0,63	0,60	0,57	0,63	0,56	0,51	0,53	0,47	0,43
4,0	0,68	0,64	0,61	0,71	0,64	0,59	0,64	0,61	0,59	0,64	0,58	0,53	0,54	0,48	0,44

За формулою (6.1) знаходимо індекс приміщення:

$$I = \frac{16 \cdot 25}{4 \cdot (16 + 25)} = 2{,}43$$

З таблиці 6.4 визначаємо значення η, за параметрами I, $\rho_{стелі}$, $\rho_{стін}$ та типом світильника.

$$\eta = 0{,}52 = 52\%$$

За формулою (6.2) визначаємо кількість світильників, при цьому відразу підставляючи значення L з формули (6.3):

$$N = \frac{16 \cdot 25}{(0.6 \cdot 4)^2} \approx 69$$

Світильники розміщуємо у чотири ряди ($N_p=4$).

За формулою (6.4) визначаємо світловий потік ламп в ряді:

$$\Phi = \frac{350 \cdot 400 \cdot 1.1 \cdot 1.6}{4 \cdot 0.52} = 118461$$

За формулою (6.5) визначаємо необхідну кількість світильників в ряду:

$$N = \frac{118461}{2 \cdot 2450} = 24.$$

Висновки

Під час розробки дипломного проєкту було систематизовано, закріплено і розширено теоретичні та практичні знання, набуті при вивченні спецдисциплін, а також здобуто вміння застосовувати отримані знання у вирішенні конкретних виробничих задач. Проведено розроблення веб-застосунку Arcane для забезпечення можливості обробки автоматизованих фінансових операцій на основі криптовалют.

Першим етапом розробки став аналіз ринку. Визначивши основних конкурентів, переваги та недоліки їхніх програмних продуктів, складено список функціональних та нефункціональних вимог до програмного продукту. В результаті аналізу є очевидний факт, що Arcane – це унікальна платформа, оскільки користувач має можливість створювати будь-який вестінг на вибір і отримувати лінійний дохід, залежно від часових рамок, за рахунок використання Arcane-токенів, при цьому накопичуючи на балансі певну суму нагород. Окрім цього, за бажанням користувач може стати потенційним партнером команди Arcane.

Загальна концепція бізнес-логіки проєкту Arcane полягає в тому, що контракт містить певні списки адрес користувачів для нарахування винагород, а також нарахування чи скасування різнотипних комісій. Користувач може купити токен і вкласти його в пул створеного вестінгу або інший криптопроєкт, який співпрацює зі створеним токеном. Фінансові операції здійснюються двома валютами: Arcane-токен та BNB. Встановлення комісій передбачено для отримання коштів на технологічні потреби та обробку транзакцій.

Оскільки Arcane-токен включає в себе інтеграцію смарт-контрактів з децентралізованої платформи PancakeSwap для точного визначення курсу валют в будь-який момент часу, прийнято рішення значно розширити функціонал смарт-контракту, передбачивши випадки можливої зміни бізнес-логіки проєкту. Додатково він може обмінювати рідний токен на основну валюту BNB мережі BscScan Testnet. При обміні за вказаним овнером в період ініціалізації контракту рутером через створену фабрику створюється пара Arcane:BNB, через яку можна здійснювати різноманітні фінансові операції не тільки з токенами проєкту, але й безпосередньо з валютою BNB.

На етапі проєктування було проведено глибокий аналіз та розроблено бізнес-логіку проєкту загалом, а також зокрема Arcane-токена, побудовано діаграму вищого рівня та виділено сценарії використання для демонстрації функціоналу смарт-контрактів, розроблено діаграму послідовності для пересилання коштів з однієї адреси на іншу за допомогою типових засобів та методів блокчейну. Після чого було визначено загальну структуру перед програмним розробленням веб-застосунку за допомогою додатку Axure RP 9 Pro Edition.

На етапі кодування було розроблено смарт-контракт ArcaneToken, протестовано його за допомогою unit-тестування, охопивши ймовірність виникнення різнотипних ситуацій. Також написано міграційні скрипти, конфігураційні файли та здійснено розгортання контракту на тестовій мережі BscScan Testnet, після чого він був додатково протестований вже на тестнеті. При написанні смарт-контракту було враховано та недопущено будь-які ризики втрати коштів, проведено рефакторинг з метою зменшення оплати за транзакції, ціна яких залежить від кількості та обсягу виконаних операцій, а також описано NatSpec-документацію перед кожним елементом у договорі. Після завершення розробки контрактів було проведено повноцінний аудит смарт-контрактів, включаючи мануальну та тестову частини, з документуванням усіх дій. Наступним кроком було розроблення front-end частини з використанням фреймворка Angular. В результаті створено зручний у використанні інтерфейс користувача з інтеграцією логіки розгорнутих контрактів.

На завершальному етапі було проведено функціональне тестування, зокрема статичне та динамічне тестування. Функціональне тестування проведено методами білої та чорної скриньки. Даний проєкт відповідає всім початковим вимогам до програмного продукту, пройшов всі тести успішно та відповідає всім функціональним та нефункціональним вимогам.

Проведено економічні дослідження для розрахунку собівартості та загальної конкурентоспроможності даного застосунку на ринку. Здійснено розрахунок заробітної плати програміста. В результаті досягнуто висновку, що застосунок є конкурентоспроможний, однак у зв’язку з дефіцитом Blockchain-розробників, а також складністю написання, перевірки та розгортання смарт-контрактів на мережах, такий децентралізований додаток є дорогим.

Під час опрацювання питань охорони праці проаналізовано та визначено потенційно небезпечні та шкідливі виробничі фактори, проведено обчислення значень освітленості приміщення, де використовується комп'ютерна техніка, а також здійснено повторне ознайомлення з правилами техніки безпеки при роботі з комп'ютерною технікою.

Отже, в результаті, отримано веб-застосунок, який відповідає всім початковим вимогам, виконує всі поставленні задачі, розроблений з використанням найсучасніших технологій та має зручний, інтуїтивно зрозумілий і ергономічний графічний інтерфейс. Окрім цього, Dapps підтримує інтеграцію умов ринку та активів інших проєктів і на основі блокчейну є цілком захищеним від несанкціонованих дій користувачів.

Список використаних джерел

1. Освітньо-професійна програма «Інженерія програмного забезпечення» від 31 серпня 2020 року. – Режим доступу до ресурсу: http://ifkepnung.if.ua/
2. Пітчук Л.В. Курсове та дипломне проєктування. Методичні вказівки. – Режим доступу до ресурсу: http://ifkepnung.if.ua/
3. What is DeFi? A beginner’s guide to decentralized finance [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://n26.com/en-eu/blog/what-is-defi
4. Блокчейн [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA%D1%87%D0%B5%D0%B9%D0%BD
5. Ethereum technology [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://ethereum.org/en/
6. Що таке смарт-контракт? [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://bankchart.com.ua/finansoviy_gid/investitsiyi/statti/scho_take_smart_kontrakt_
7. Введення в Binance Smart Chain (BSC) [Електронний ресурс] // 2020. – Режим доступу до ресурсу: https://academy.binance.com/uk/articles/an-introduction-to-binance-smart-chain-bsc
8. MetaMask [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://metamask.io/
9. Що таке Solidity та як він використовується для розумних контрактів? - Посібник Лаймана [Електронний ресурс] // 2018. – Режим доступу до ресурсу: https://www.newgenapps.com/uk/blogs/what-is-solidity-meaning-uses-ethereum-virtual-machine/
10. Бібліотека OpenZeppelin [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://docs.openzeppelin.com/
11. Hardhat [Електронний ресурс] // 2022 – Режим доступу до ресурсу: https://hardhat.org/
12. Truffle [Електронний ресурс] // 2022 – Режим доступу до ресурсу:

https://trufflesuite.com/docs/truffle/testing/testing-your-contracts/

13. BscScan: Introduction [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://docs.bscscan.com/

14. HTML: HyperText Markup Language [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTML

15. Component styles of Angular [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://angular.io/guide/component-styles

16. TypeScript Tutorial Info [Електронний ресурс] // 2019. – Режим доступу до ресурсу: https://codeguida.com/post/475

17. JavaScript Info [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

18. AngularJS – Wiki [Електронний ресурс] // 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS

19. Applicature Universal Components [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://www.npmjs.com/package/@applicature/components

20. Transparent Upgradeable Proxy [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://docs.openzeppelin.com/contracts/3.x/api/proxy#TransparentUpgradeableProxy

21. Events in Solidity language [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://www.tutorialspoint.com/solidity/solidity_events.htm

22. Застосування UML. Діаграма послідовності [Електронний ресурс] // 2020. – Режим доступу до ресурсу: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy

23. AVADA-MEDIA [Електронний ресурс] // 2022. – Режим доступу до ресурсу: https://avada-media.ua/ua/services/razrabotka-smart-kontrakta-dlya-bizneca/

24. Q & A / Навчальний ресурс з тестування програмного забезпечення [Електронний ресурс] // 2018. – Режим доступу до ресурсу: https://qalearning.com.ua/theory/lectures/material/testing-levels/

25. Ручне та автоматизоване тестування [Електронний ресурс] // 2018. – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/

26. Охорона праці при роботі з ПК [Електронний ресурс] // 2020. – Режим доступу до ресурсу: https://www.sop.com.ua/article/183-ohoron-prats-pri-robot-z-kompyuterom

27. Шкідливі та небезпечні виробничі чинники при роботі з комп'ютерною технікою [Електронний ресурс] // 2021. – Режим доступу до ресурсу: https://studfile.net/preview/5211197/page:3/

28. Вимоги до освітлення [Електронний ресурс] // 2021. – Режим доступу до ресурсу: https://studfile.net/preview/5211197/page:9/

29. Захисне заземлення [Електронний ресурс] // 2021. Режим доступу до ресурсу: https://studfile.net/preview/5211197/page:13/