Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг»

Виконав: студент 2-го курсу групи 1ІСТ-22м
спеціальності 126 – Інформаційні системи та технології
Володимир ЖИГАНОВ

Керівник: д.т.н., проф. каф АІІТ
Роман КВЄТНИЙ

Консультант: к.т.н., доц.каф.АІІТ
Ілона БОГАЧ
«11» грудня 2023 р.

Опонент: д.т.н. доцент КН
Олексій СІЛАГІН
«12» грудня 2023 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО.
«14» грудня 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 12 – Інформаційні технології

Спеціальність – 126 Інформаційні системи та технології

Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО.

«20» 09 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Жиганову Володимиру Андрійовичц

(ПІБ автора повністю)

1. Тема роботи: Розробка архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг

   Керівник роботи: д.т.н., проф. каф. АІІТ Роман КВЄТНИЙ

   Затверджені наказом ВНТУ від «18» 09 2023 року №247.
2. Строк подання студентом роботи до «20» листопада 2023 року.
3. Вихідні дані до роботи: операційна система Windows 10 або Ubuntu 16 і вище; доступ до інтернету; оперативна пам'ять 4Гб; процесор intel core i5.
4. Зміст текстової частини: вступ; загальна характеристика процесів розробки частин веб застосунку на основі технології блокчейн; проектування компонентів та підходи до розробки програмного забезпечення; розробка інтерфейсу та процеси забезпечення якості веб-застосунку; економічна частина; висновки; список використаних джерел
5. Перелік ілюстративного (або графічного) матеріалу: схема взаємодій користувача згідно вимог; схема загальної роботи застосунку; схема загальної роботи плагіну, вигляд основних сторінок додатку.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Ім'я ПРІЗВИЩЕ та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийн
1-4	Ілона БОГАЧ, к.т.н., доцент каф. АІТ		
5	Володимир КОЗЛОВСЬКИЙ, к.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання «_20_»___09___2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	20.09.23-03.10.23	
2	Огляд існуючих програмних засобів для реалізації поставленої задачі. Вибір оптимальних інформаційних технологій	03.10.23-15.10.23	
3	Вибір мови програмування та середовища розробки. Проектування системи	15.10.23-20.10.23	
4	Програмна реалізація. Експериментальна перевірка роботи програми	20.10.23-14.11.23	
5	Економічна частина	14.11.23-16.11.23	
6	Оформлення пояснювальної записки і графічного матеріалу	17.11.23-20.11.23	
7	Попередній захист роботи	21.11.23	
8	Остаточний захист роботи	10.12.23– 20.12.23	

Студент Володимир ЖИГАНОВ

Керівник роботи Роман КВЄТНИЙ

АНОТАЦІЯ

УДК 004.75

Жиганов В. А. Розробка архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 107с.

На укр. мові. Бібліогр.: 39 назв; рис.: 31; табл.: 17.

У магістерській кваліфікаційній роботі досліджено підходи до розробки веб-застосунків, що використовують технології та засоби блокчейн мереж. Також, звертається увага на собливості підходів до тестування застосунків, що використовують блокчейн-технології. Розроблений графічний інтерфейс та інфраструктура веб-застосунку надають безперервності в роботі користувача з додатком та надання можливості розробникам впроваджувати нові зміни.

Ключові слова: блокчейн, тестування блокчейн мереж, пайплайн, графічний інтерфейс, програмне забезпечення.

ABSTRACT

Zhyhanov V.A. The development of architecture, interface, and product quality assurance for integrating blockchain technology into the service industry. Master's qualification paper of a specialty 126 – Informational systems and technologies, educational program – Informational technologies of data and image analysis. Vinnytsia: VNTU, 2023. 107p.

In Ukrainian language. Bibliography: 39 titles; fig.: 31; tabl.: 17.

In the master's thesis, approaches to developing web applications using blockchain technologies and tools were investigated. Attention is also paid to the peculiarities of testing applications that utilize blockchain technologies. The developed graphical interface and web application infrastructure provide continuous user interaction with the application and enable developers to implement new changes.

Keywords: blockchain, testing blockchain networks, pipeline, graphical interface, software.

ЗМІСТ

ВСТУП 4
1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРОЦЕСІВ РОЗРОБКИ ЧАСТИН ВЕБ-ЗАСТОСУНКУ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН 6
1.1 Технологія Blockchain. Основні поняття 6
1.2 Основні компоненти блокчейну 7
1.3 Процеси менеджменту проєкту та розробки на основі технології блокчейн 8
1.4 Аналіз існуючих веб-застосунків, які використовують технологію блокчейн .. 10
1.5 Аналіз інструментарію та процесів розробки веб-застосунку 14
1.6 Висновки до розділу 17
2 ПРОЕКТУВАННЯ КОМПОНЕНТІВ ТА ПІДХОДИ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 18
2.1 Життєвий цикл розробки програмного забезпечення для блокчейн-застосунків 18
2.2 Опис вимог до функціональних можливостей застосунку 20
2.3 Основні компоненти блокчейн-застосунків 30
2.3.1 Характеристики блокчейну та типи технології 30
2.3.2 Смарт-контракти на основі блокчейну 34
2.3.3 Блокчейн Ethereum 34
2.3.4 Solidity 35
2.3.5 Metamask 37
2.4 Обгрунтування вибору програмного забезпечення 38
2.5 Порівняння сучасних підходів до розробки та тестування веб-застосунків із запропонованим методом тестування 42
2.6 Висновки до розділу 44
3 РОЗРОБКА ІНТЕРФЕЙСУ ТА ПРОЦЕСИ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ВЕБ-ЗАСТОСУНКУ 46
3.1 Розробка інтерфейсу 46
3.1.1. Підхід до розробки інтерфейсу 46

3.1.2. Опис технологій, використаних в розробці інтерфейсу 48
3.1.3 Опис головних компонентів системи 49
3.2 Послідовність кроків в процесі розробки програмного забезпечення 51
3.3 Характеристика процесів забезпечення якості веб застосунку 54
3.3.1 Планування тесового процесу 54
3.3.2 Види тестування систем 56
3.3.3 Інструменти для тестування блокчейн веб-застосунків 60
3.3.4 Аналіз процесів виявлення та виправлення помилок 61
3.4 Автоматизація тестування блокчейн-застосунку. 64
3.5 Аудит смарт-контрактів 65
3.6 Висновок до розділу 68
4 ЕКОНОМІЧНА ЧАСТИНА 69
4.1 Технологічний аудит розробленого веб застосунку 69
4.2 Розрахунок витрат на розроблення мобільного додатка та програмного інтерфейсу 74
4.3 Розрахунок економічного ефекту від можливої комерціалізації розробки 79
ВИСНОВКИ 88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 89
Додатки 93
Додаток А (обов’язковий) Технічне завдання 94
Додаток Б (обов’язковий) Ілюстративна частина 97
Додаток В (обов’язковий) Лістинг розроблених тестів 101
Додаток Г Довідка про впровадження результатів роботи 106
Додаток Д (обов’язковий) Протокол перевірки на плагіат 107

ВСТУП

Актуальність роботи. Технології блокчейну активно розвиваються у сучасному світі [1]. Зростання інтересу до цієї технології спричинене її ключовими особливостями: підвищеною безпекою даних, децентралізацією, стійкістю до змін або видалення інформації, а також швидким здійсненням транзакцій між учасниками мережі. Блокчейн є ідеальним рішенням для задач, де найважливішими є надійність та конфіденційність даних, які гарантуються завдяки криптографічним методам та його децентралізованій структурі [2].

Велика кількість підприємців, організацій та сервісних провайдерів використовують онлайн-додатки, щоб збільшити кількість клієнтів. Однак, існує значний ризик щодо безпеки та надійності онлайн-сервісів. З моменту появи Інтернету, шахрайство розвивалося паралельно з технологічним прогресом, і шахраї не припиняють шукати нові методи вторгнення. Однак блокчейн мережі є менш вразливими до таких загроз. Тому технологія блокчейну стала популярною в середовищі сервісних послуг, зокрема у фінансовій сфері. Децентралізовані програми (DAPP [3]) базуються на смарт-контрактах в evm-сумісній мережі, що гарантує безпеку даних користувачів. [4-9]

Актуальністю роботи є необхідність покращення процесу надання послуг та підвищення рівня їх надійності з використанням сучасних технологій, що може бути використаний при створенні або інтеграції в уже існуючі онлайн платформи для надання послуг продажу та покупок.

Метою магістерської кваліфікаційної роботи є покращення процесу забезпечення якості продуктів у сфері послуг, що використовують технологію блокчейн, за рахунок використання сучасних технологій, що дозволить суттєво спростити процеси тестування та покрити всі користувацькі сценарії.

Об'єктом досліджень є процеси розробки та забезпечення якості систем для надання послуг продажу та покупок з використанням блокчейн технологій як механізмом збереження, обробки та валідації даних.

Предметом досліджень є методи та засоби тестування застосунків, які використовують блокчейн технології.

Для досягнення мети необхідно розв'язати наступні задачі:

1. Вивчення мережі блокчейн та її можливостей для аналізу способів використання для досягнення поставленої мети.

2. Дослідити різні програмні рішення надання послуг, де можна застосувати блокчейн технології.

3. Проектування програмної частини додатку.

4. Розробити графічний інтерфейс та забезпечити покриття всіх користувацьких сценаріїв.

5. Провести тестування додатку, аудит смарт-контрактів та розробити тестову документацію для покриття модулів додатку та порівняти з аналогами.

Новизна отриманих результатів роботи полягає в тому, що запропонована модифікація процесу забезпечення якості продуктів у сфері послуг, які використовують технологію блокчейн, що на відміну від існуючих аналогів дозволяє використати всі етапи аудиту, суттєво спростити процеси тестування та покрити всі користувацькі сценарії.

Практична цінність отриманих результатів полягає в розробці додатку з відкритим кодом [10] та якісно-документованому API, що застосовується як окремий сервіс, що може бути використаний для тестування при створенні або інтеграції в уже існуючі онлайн платформи для надання послуг.

Апробація результатів дослідження: результати, отримані в магістерській кваліфікаційній робот опубліковані в матеріалах щорічної всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, ВНТУ, 2023 р) [11].

Впровадження результатів роботи.

Результати роботи були впровадженні в розробки підприємства ТОВ «ФАЙВ Системз Девелопмент» (Додаток Г). Основний модуль роботи прийнятий *на отримання свідоцтва авторського права на твір.*

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРОЦЕСІВ РОЗРОБКИ ЧАСТИН ВЕБ-ЗАСТОСУНКУ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

1.1 Технологія Blockchain. Основні поняття

Блокчейн – це інноваційна технологія, яка за короткий час стала справжнім проривом в інформаційних технологіях. Він представляє собою розподілений реєстр, де інформація зберігається в блоках, які поєднані між собою. Основні характеристики блокчейну включають децентралізованість, високий рівень безпеки і прозорість. Ця технологія вже знайшла своє застосування у багатьох галузях, від фінансів до охорони здоров'я.

Кожен блок в мережі містить інформацію, яка шифрується за допомогою криптографічних хешів. Ці блоки з'єднані між собою, створюючи ланцюг. Ключовим моментом є досягнення консенсусу між учасниками мережі щодо правильності даних. А смарт-контракти дозволяють автоматизувати та безпосередньо виконувати угоди без посередників.

Децентралізація виступає однією з основних характеристик технології блокчейн. Відмінно від звичних централізованих систем, де існує один контролюючий орган або посередник, що регулює доступ і дії, блокчейн пропонує структуру без єдиного центру керівництва. Всі дані і функції управління розпределені серед усіх вузлів мережі, надаючи кожному учаснику можливість володіти повним дублікатом всієї бази інформації. Така структура сприяє підвищенню прозорості, автономності та безпеки платформи [12].

В блокчейн технології ключовим елементом є принцип консенсусу, який передбачає згоду між учасниками мережі щодо статусу бази даних. У традиційних системах такий процес контролюється одним центром, а в блокчейні вибір робиться на основі згоди учасників. Існує декілька методів досягнення консенсусу, таких як Proof of Work (PoW) або Proof of Stake (PoS), які забезпечують узгодженість учасників щодо вірності блоків і операцій.

Технологія блокчейн відзначається великою мірою безпеки завдяки своїм

унікальним характеристикам. Так як інформація розсіюється по різних вузлах мережі, модифікація або порушення вже існуючих блоків стає вкрай складним. Більше того, кожна дія підтверджується усією мережею через механізми, такі як майнінг (для PoW) або стейкінг (для PoS), роблячи систему стійкою до недобросовісних дій [13].

Для ефективного застосування блокчейну важливо розуміти його основні концепції. Принципи, такі як децентралізація, консенсус та надійність, є стовпами технології блокчейн, які сприяють створенню прозорої, безпечної та надійної системи обміну даними. Використовуючи ці засади, блокчейн пропонує величезний потенціал для створення новітніх веб-рішень та відповідей на актуальні виклики в різних галузях.

1.2 Основні компоненти блокчейну

Основна структурна одиниця блокчейну є блок. Він містить пакет транзакцій, які були виконані в мережі протягом певного часового проміжку. Кожний блок також має заголовок, який містить метадані, такі як час створення блоку та посилання на попередній блок у ланцюзі.

Щоб забезпечити інтегральність даних, кожен блок містить унікальний криптографічний хеш-код, який генерується на основі інформації у блоку. Якщо хоча б одна деталь у блоку зміниться, хеш також зміниться, і це буде легко виявлено.

Коли новий блок створюється, він додається до ланцюга блоків, з'єднуючись з попереднім блоком за допомогою його хешу. Це створює неперервний ланцюг від першого блоку (генезис-блоку) до останнього.

Для забезпечення довіри до системи та підтвердження правильності транзакцій, у блокчейні використовуються алгоритми консенсусу. Ці алгоритми забезпечують, що всі учасники мережі згодні з доданими в мережу даними.

Основним джерелом інструкцій в мережі блокчейн є смарт-контракт. Смарт-контракти – це самовиконувані програми, які діють на основі блокчейну.

Вони автоматично активуються, коли задані умови зустрічаються.

Основна перевага смарт-контрактів полягає в їхній здатності автоматизувати процеси, роблячи транзакції більш ефективними. Оскільки вони базуються на блокчейн-технології, смарт-контракти надійно захищені від зовнішніх втручань та шахрайства.

Традиційно, більшість угод потребує участі посередників, таких як банки, нотаріуси або інші треті сторони, що може збільшувати вартість та час виконання угоди. Смарт-контракти видаляють цю необхідність, забезпечуючи безпосередні транзакції між сторонами.

Смарт-контракти можуть бути дуже гнучкими та адаптованими до різних сценаріїв. Вони можуть включати різні логічні умови, від простих до дуже складних, в залежності від конкретних потреб користувача.

Оскільки вони зберігаються на блокчейні, всі учасники мережі можуть перевірити деталі смарт-контракту, що забезпечує високий рівень прозорості.

Однак, не дивлячись на їхні переваги, смарт-контракти також мають свої виклики. Розробка надійного смарт-контракту вимагає глибоких знань та спеціалізованого досвіду. Крім того, якщо в смарт-контракті є помилки або уразливості, це може призвести до втрати коштів або інших проблем.

1.3 Процеси менеджменту проєкту та розробки на основі технології блокчейн

Впровадження технології блокчейн у розробку веб-застосунків вимагає детального розгляду кожного етапу проєкту. Етапи розробки починаються з аналізу бізнес-вимог та визначення функціональності системи, враховуючи особливості блокчейн технології. Проектування архітектури повинно включати в себе розробку смарт-контрактів та визначення структури децентралізованої бази даних.

Управління версіями та конфігураційні процеси стають важливими на

етапі програмування. Забезпечення цілісності коду через системи контролю версій, такі як Git, та ефективне управління конфігураціями дозволяє вирішувати супутні завдання блокчейн розробки.

Паралельно з розробкою системи, також виконуються процеси забезпечення якості та тестування. Виділення ключових сценаріїв користувача, головних компонентів системи для розробки тестової документації, яка надає чітку і якісну картину реального стану продукту та його готовності до використанням користувачем. Окрім того, впровадження автоматизованого тестування та його виконання під наглядом інженера із забезпечення якості (Quality Assurance engineer) допомагає економити час та ресурси на проходження ключових сценаріїв користувача, а також перевірок працездатності всіх модулів системи. Однією із особливостей розробки блокчейн застосунків – наявність смарт-контрактів. Тестування смарт-контрактів вимагає створення тестових кейсів для перевірки логіки контрактів та автоматизації процесу за допомогою інструментів, таких як Truffle або Remix. Одночасно, тестування застосунку вцілому включає інтеграційні тести, тести безпеки та використання тестових мереж для емуляції реальних умов.

У контексті проектів, пов'язаних із розробкою веб-застосунків на основі технології блокчейн, управління ресурсами та розподіл обов'язків є визначальними аспектами, що впливають на загальний результат. Успішна реалізація вимагає чіткого розподілу обов'язків між членами команди. Кожен учасник повинен мати чітко визначену сферу відповідальності, щоб уникнути дублювання робіт та конфліктів. Ефективний розподіл обов'язків досягається за допомогою методологій розробки, таких як Scrum чи Kanban. Scrum надає гнучкий підхід до розподілу робіт на невеликі ітерації, а Kanban сприяє візуалізації робочого процесу та керуванню завданнями на дошці завдань. Інструменти для управління проектами, такі як Jira чи Trello, стають невід'ємною частиною управління ресурсами та обов'язками. Вони дозволяють створювати, відстежувати та пріоритизувати завдання, а також забезпечують можливість взаємодії між членами ко-

манди. Управління ресурсами також включає в себе забезпечення команди необхідними інструментами та технічними ресурсами. Це може охоплювати навчання членів команди, використання спеціалізованих програмних засобів та забезпечення необхідного апаратного забезпечення. Чітке визначення ролей, застосування методологій та використання ефективних інструментів дозволяють забезпечити якісну та ефективну розробку веб-застосунків на основі технології блокчейн.

1.4 Аналіз існуючих веб-застосунків, які використовують технологію блокчейн

Блокчейн технологія, як розподілена система зберігання та передачі даних, знаходить все більше застосувань у веб-середовищі. Ця інноваційна технологія надає нові можливості для створення безпечних веб-застосунків.

Блокчейн активно впроваджують в різні сфери діяльності людини, в якій є робота з інформацією. Такі галузі як економіка, зберігання та обмін даних, медицина, соціальні мережі, маркетплейси для торгівлі віртуальними речами та ін.

Blockchain.info є однією з провідних платформ, спеціалізованих на наданні сервісів у сфері криптовалют. Сайт розроблено таким чином, щоб користувачам було зручно та безпечно управляти своїми криптовалютними активами. Основною перевагою платформи є її простота та інтуїтивність. Всі інструменти для управління валютою, такі як створення гаманця, відправлення та отримання коштів, а також перевірка балансу, розміщені на зручних панелях.

Окрім того, однією з ключових особливостей Blockchain.info є можливість перевірки історії транзакцій. Користувачі можуть відстежувати всі свої транзакції, дивитися деталі кожної з них, а також перевіряти статус підтвердження. Ця функція особливо корисна для тих, хто хоче мати повний контроль над своїми фінансами та бути в курсі всіх змін на своєму балансі.

Також варто відзначити, що Blockchain.info піклується про безпеку своїх

користувачів. Платформа використовує різноманітні технології захисту, включаючи двофакторну аутентифікацію, щоб забезпечити конфіденційність та безпеку коштів своїх користувачів.

MetaMask – це криптографічний гаманець та веб-розширення для браузерів, яке стає мостом між звичайними веб-браузерами та блокчейном Ethereum. З його допомогою користувачі можуть взаємодіяти з децентралізованими застосунками (DApps) на Ethereum без завантаження повного вузла блокчейна. Він доступний для таких браузерів, як Chrome, Firefox, Brave та Edge. Після встановлення користувачі можуть створювати нові гаманці або імпортувати існуючі, при цьому їх приватні ключі завжди зберігаються локально на пристрої користувача. MetaMask забезпечує можливість легко взаємодіяти з різними DApps, виконувати транзакції, відправляти та отримувати Ether та інші токени стандартів ERC-20 та ERC-721. Цей гаманець також дозволяє легко перемикатися між різними мережами Ethereum, включаючи тестові мережі. Окрім цього, для підвищення безпеки, MetaMask пропонує такі функції, як двофакторна аутентифікація та автоматичне блокування після декількох невдалих спроб входу. Однією з нових функцій є "Swaps", яка дозволяє користувачам безпосередньо обмінювати токени в межах гаманця. Загалом, MetaMask є ключовим елементом екосистеми Ethereum, який забезпечує зручність і безпеку при роботі з блокчейном [14].

Filecoin - це децентралізована мережа, яка дозволяє користувачам зберігати, обмінюватися та отримувати доступ до файлів. Замість традиційного централізованого сховища даних, Filecoin використовує блокчейн для зберігання файлів. Користувачі можуть надійно зберігати свої файли, а також використовувати протокол Filecoin для передачі цих файлів іншим користувачам у мережі.

Hive є децентралізованою платформою для соціальних медіа на основі технології блокчейн. Основна ідея Hive полягає в повній децентралізації, що дозволяє забезпечити гнучкість та відкритість платформи, при цьому відсутність контролю з боку будь-якого окремого органу чи групи. Користувачі можуть публікувати різноманітний контент, включаючи статті, відео та графіку. Інші учасники мережі можуть висловлювати своє ставлення до публікацій за допомогою

голосування, а найбільш популярні пости отримують винагороди у вигляді криптовалюти Hive. Спільнота є ключовим аспектом Hive, і завдяки відкритому коду та прозорості операцій, користувачі активно беруть участь у процесах прийняття рішень. Множинність додатків та інструментів, які були створені завдяки відкритому API Hive, збагачують екосистему платформи, пропонуючи користувачам блоги, ігри та інші сервіси. Блокчейн технологія, на якій базується Hive, забезпечує високий рівень безпеки транзакцій та даних користувачів. Крім того, Hive пропонує різні способи монетизації для своїх користувачів, включаючи винагороди за контент, торгівлю криптовалютою та участь у рекламних кампаніях. Усі ці особливості роблять Hive цікавою та перспективною платформою в світі соціальних медіа на блокчейні.

OpenSea — це онлайн-ринок для NFT (невидалинних токенів), який використовує технологію блокчейн для створення, продажу та обміну цифрових мистецьких та ігрових активів. Платформа використовує технологію смарт-контрактів для створення, продажу та обміну цифрових активів, які можна ідентифікувати за допомогою токенів. OpenSea підтримує стандарт ERC-721, який визначає унікальні та необмінювані токени. Користувачі можуть створювати, купувати та продавати NFT безпосередньо на платформі. Унікальність цифрових активів і можливість їх безпечного обміну зробили OpenSea важливим гравцем у ринку цифрового мистецтва та ігрових активів на блокчейні [15].

Modum - це стартап, що базується в Швейцарії, який пропонує систему на базі блокчейн для відстеження фармацевтичних товарів. Зокрема, Modum спеціалізується на відстеженні та забезпеченні відповідного зберігання лікарських препаратів під час їх перевезення, щоб гарантувати, що температурний режим завжди відповідає стандартам. Це допомагає запобігти псуванню препаратів та забезпечує їх якість до моменту отримання кінцевим споживачем. Технологія блокчейн в Modum дозволяє створити незмінний ланцюжок даних про умови перевезення кожного конкретного товару, що підвищує довіру до продукції та забезпечує прозорість для всіх сторін логістичного ланцюжка.

SushiSwap - це децентралізована біржа на базі Ethereum, яка, подібно до

Uniswap, використовує автоматичні маркет-мейкери (AMM) для децентралізованого обміну токенів. Проте, SushiSwap відрізняється від Uniswap додатковими функціями та унікальною екосистемою управління спільнотою. Однією з ключових особливостей SushiSwap є їх токен SUSHI, який не лише дає винагороди за надання ліквідності, але й дозволяє його власникам брати участь в управлінні платформою. SushiSwap був створений як форк Uniswap, але з метою створення більш демократичної структури та введення додаткових винагород для учасників спільноти.

Багато організацій вдаються до технології блокчейн під час створення вебдодатків через численні переваги, які вона пропонує. До таких переваг належать децентралізованість, відкритість, захист інформації, стабільність та відсутність непотрібних посередників. Здійснення операцій з активами стає простішим завдяки системам автоматичного обміну токенами, як-от Uniswap. Смарт-контракти надають змогу автоматизувати угоди та корпоративні операції.

Крім того, блокчейн забезпечує універсальний доступ та надійність маленьких платежів. У фінансовій галузі ця технологія сприяє безпечності та оперативності монетарних операцій. У сфері нерухомості додатки, такі як Propy, застосовують блокчейн для надійного передачі прав власності. А в медичній галузі, MediLedger допомагає контролювати потік медичних препаратів протягом всього ланцюга постачання.У мистецтві та ігровій індустрії, технологія блокчейну дозволяє створювати та обмінювати унікальні цифрові активи, як NFT. Системи віртуального голосування, наприклад Voatz, забезпечують безпеку та надійність. Інновації в голосуванні та додатки для IoT [16] також використовують блокчейн для забезпечення безпеки та надійності. Крім того, громадські реєстри, зменшення шахрайства та розвиток технологій грають важливу роль у популяризації цієї технології.

Отже, технологія блокчейн широко використовується в розробці веб-застосунків через її низку переваг. Вона революціонізує підходи до децентралізації, прозорості, безпеки даних та надійності. Ліквідність активів, мікроплатежі та ефективні смарт-контракти стають реальністю завдяки цій технології.

Глобальний доступ, безпека транзакцій, інновації в голосуванні та відстеженні постачання ліків — це лише частина областей застосування. У сфері нерухомості, мистецтва, медицини та ігор блокчейн відкриває нові можливості та забезпечує надійні та безпечні технологічні рішення. Його вплив на розвиток цифрового світу надто важливий, і ймовірно, тільки початок його використання у різних галузях технологій.

1.5 Аналіз інструментарію та процесів розробки веб-застосунку

Розробка веб-застосунків, які використовують технологію блокчейн, вимагає специфічного підходу, знань та компетенцій, які відрізняються від традиційної веб-розробки.

Перед тим, як розпочати розробку, важливо провести глибокий аналіз ринку, щоб зрозуміти, які конкретні потреби може задовольнити даний продукт. Окрім визначення основних функцій та цільової аудиторії, рекомендується провести опитування потенційних користувачів або провести тестовий запуск MVP (мінімально життєздатний продукт) для отримання зворотного зв'язку. Основуючись на цих даних, можна коригувати вимоги до продукту, зокрема, з урахуванням особливостей блокчейну [17].

Після визначення вимог наступний крок — це планування архітектури. Оскільки блокчейн часто вимагає високої пропускної здатності та швидкого відгуку на запити, важливо ретельно підібрати серверне рішення та базу даних. Також потрібно враховувати можливість масштабування застосунку зі зростанням кількості користувачів.

При інтеграції з блокчейном, важливо звернути увагу на вибір конкретного блокчейну (наприклад, Ethereum, Binance Smart Chain чи інші) та розглянути варіанти використання власного вузла (node) чи звертання до сторонніх сервісів для взаємодії з блокчейн мережею.

Щодо вибору мови програмування, деякі платформи блокчейн пропонують специфічні мови або фреймворки для розробки смарт-контрактів, такі як

Solidity для Ethereum. Отже, команда розробників повинна мати відповідні знання та навички для ефективної роботи.

Невід'ємною частиною розробки програмного забезпечення є тестування. Тестування веб-застосунків, інтегрованих з технологією блокчейн, має свої особливості, які відрізняють його від стандартних методів тестування. Одна з ключових відмінностей полягає в незмінності блокчейну. Це означає, що після додавання даних до блокчейну їх вже не можна змінити чи видалити. Ця особливість робить традиційне регресійне тестування менш придатним. Крім того, блокчейн-технологія вимагає тестування в умовах децентралізованої системи, що є відмінною від звичних централізованих систем.

З огляду на високі вимоги до доступності блокчейну, виникає необхідність проведення додаткового тестування продуктивності і надійності. Окрема увага приділяється тестуванню смарт-контрактів. Ці автоматизовані контракти виконують дії на основі певних умов, і будь-яка помилка в їх коді може призвести до великих втрат.

Одним з найпопулярніших інструментів для розробки і тестування смарт-контрактів на Ethereum є Truffle. Цей фреймворк надає потужний набір засобів, що дозволяє розробникам писати, тестувати та розгортати смарт-контракти. Він також має вбудовані бібліотеки для автоматизації тестування та проведення міграцій. З його допомогою можна легко імітувати реальні умови роботи смарт-контракту, включаючи інтеракцію з іншими контрактами та взаємодію з Ethereum Virtual Machine (EVM).

Ganache, з іншого боку, є частиною екосистеми Truffle і виступає у якості локального блокчейн-сервера для розробки. Цей інструмент дозволяє розробникам створювати власні приватні тестові мережі, що імітують реальний блокчейн Ethereum. На рисунку 1.1 зображено початковий екран застосунку.

Головна перевага Ganache полягає в тому, що розробники можуть проводити всі види тестувань та експериментів без ризику впливу на реальний блокчейн та витрат ефірів. Крім того, Ganache надає корисний графічний інтерфейс,

де можна переглядати деталі транзакцій, стан смарт-контрактів та іншу інформацію про блокчейн.

Окрім тестування, важливим етапом є аудит смарт-контрактів. Аудит смарт-контрактів є критично важливим кроком у процесі розробки, оскільки він забезпечує додатковий рівень захисту для користувачів і гарантує, що код виконується відповідно до заявлених специфікацій. Цей процес спрямований на виявлення і виправлення потенційних вразливостей, помилок або недоліків у коді, що можуть призвести до втрати коштів або інших небажаних наслідків для користувачів.

Mythril є одним з передових інструментів для проведення динамічного аналізу безпеки смарт-контрактів. Він автоматично виявляє вразливості, аналізуючи байт-код контракту і моделюючи його поведінку. Mythril також надає докладні звіти, які допомагають розробникам зрозуміти потенційні проблеми та шляхи їх вирішення.

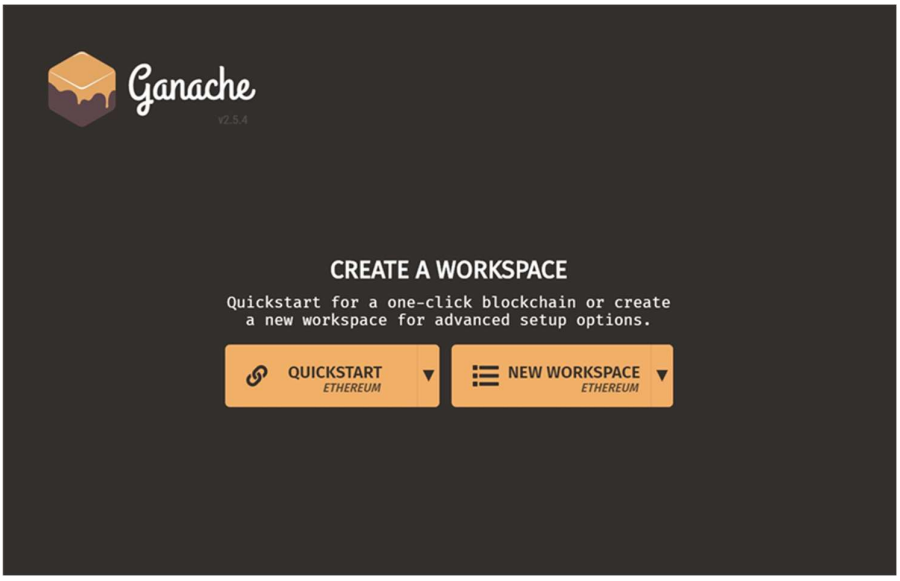


Рисунок 1.1 – Початковий екран застосунку Ganache

Slither, з іншого боку, є інструментом для статичного аналізу коду смарт-контрактів. Він перевіряє вихідний код на наявність відомих вразливостей, пате-

рнів ненадійного коду або інших потенційних проблем. Завдяки глибокому аналізу коду, Slither може виявити тонкі помилки або небезпечні патерни, які можуть бути пропущені під час ручного перегляду.

Обидва цих інструменти взаємодоповнюють один одного і, коли вони використовуються разом, надають глибоке розуміння стану безпеки смарт-контракту. Важливо розуміти, що автоматичний аудит - це лише частина процесу; ручний перегляд коду досвідченими аудиторами завжди буде важливим для забезпечення найвищого рівня безпеки. В цілому, інтеграція та тестування блокчейн-застосунків вимагає глибокого розуміння особливостей технології, а також використання спеціалізованих інструментів для забезпечення якості та безпеки рішень.

1.6 Висновки до розділу

Розглядаючи загальну характеристику процесів розробки частин веб-застосунку на основі технології блокчейн, важливо відзначити, що сам проект, який передбачає розробку архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг, є унікальним і не має наявних робочих аналогів. Використання блокчейн у веб-застосунках визначає нові стандарти та вимоги до процесів розробки.

За результатами аналізу було визначено, що децентралізація, стійкість до несанкціонованих змін і швидкість транзакцій, які є ключовими характеристиками технології блокчейн, впливають на всі етапи розробки, від планування до тестування.

2 ПРОЕКТУВАННЯ КОМПОНЕНТІВ ТА ПІДХОДИ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Життєвий цикл розробки програмного забезпечення для блокчейн-застосунків

Життєвий цикл програмного забезпечення (рис. 2.1) є комплексним процесом, що охоплює всі етапи життєвого циклу від постановки завдань до виведення продукту з експлуатації.

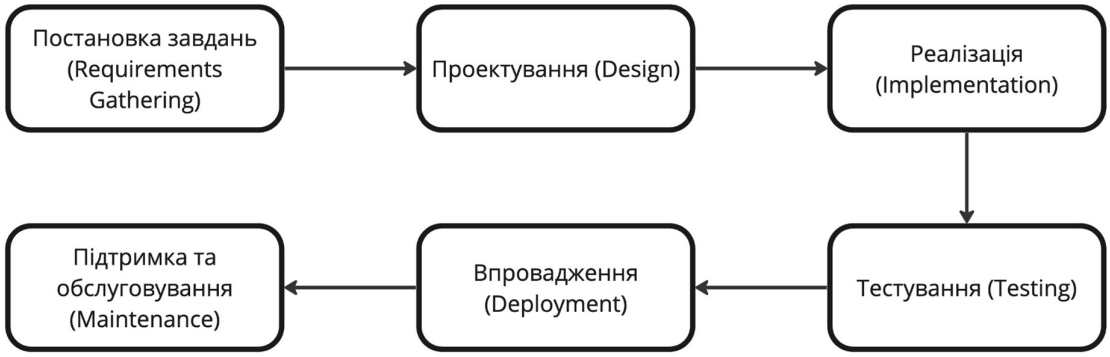


Рисунок 2.1 - Життєвий цикл програмного забезпечення

Першочергово, цей цикл визначає стратегію розробки, враховуючи вимоги клієнта та конкретні особливості проекту.

На етапі планування формуються концепції продукту, визначаються ресурси, терміни та бюджет. Розробка включає етапи проектування, реалізації коду та тестування. Завдяки цьому, можна уникнути багатьох проблем на ранніх етапах, а сам продукт буде забезпечений високою якістю та відповідатиме стандартам.

Впровадження передбачає реліз та поширення програмного продукту, а підтримка — його подальшу оптимізацію та виправлення помилок протягом експлуатації. Життєвий цикл дозволяє виявляти та виправляти проблеми на кож-

ному етапі, забезпечуючи ефективність процесу розробки та довгостроковий успіх продукту [18].

Постановка завдань (Requirements Gathering): На цьому першому етапі розробки програмного продукту вирішується фундаментальне питання: "Що саме має вирішувати цей продукт?". Команда ретельно вивчає потреби клієнта, проводить аналіз бізнес-процесів та визначає як функціональні, так і нефункціональні вимоги до програми. Цей етап включає в себе комунікацію з різними зацікавленими сторонами, від замовників до кінцевих користувачів, щоб зрозуміти всі аспекти очікувань від майбутнього продукту. Результатом є документ, який чітко формалізує вимоги до програми.

На етапі проектування (Design), визначені на попередніх стадіях концепції та вимоги перетворюються в конкретні архітектурні рішення та дизайн програмного продукту. Цей етап включає в себе визначення структури, компонентів та інших важливих аспектів системи. Також на цьому етапі вибираються технології, які найбільш відповідають поставленим завданням, що дозволяє створити план реалізації, описуючи взаємодію різних компонентів програми.

Реалізація (Implementation) - це етап, який настає після визначення дизайну, коли розробники переходять до написання самого програмного коду. На цій стадії концепції та ідеї перетворюються в функціональність продукту. Процес реалізації включає програмування, тестування окремих частин коду (юніт-тестування) та ітеративні коригування. Розробники активно втілюють задумане, роблячи функціональні частини програми працездатними та відповідними вимогам проекту.

Етап тестування (Testing) включає в себе проведення різноманітних видів тестів для забезпечення якості продукту. Юніт-тести перевіряють окремі частини коду, інтеграційні тести — взаємодію компонентів, системні тести — роботу всієї системи. Цей етап також включає в себе валідацію продукту відповідно до визначених вимог.

Впровадження (Deployment): Розроблений продукт готовий для використання та впровадження в робоче середовище. На цьому етапі здійснюється реліз програми, проводяться необхідні заходи з підготовки користувачів, і можливо, створюються інструкції для кінцевих користувачів.

Підтримка та обслуговування (Maintenance): Цей етап включає в себе надання технічної підтримки для користувачів, виправлення будь-яких помилок, які можуть виникнути в експлуатації, і вдосконалення продукту відповідно до нових вимог. Також може включати в себе розширення функціоналу для відповіді на зміни в бізнес-середовищі. Підтримка та обслуговування важливі для тривалого та успішного використання продукту.

2.2 Опис вимог до функціональних можливостей застосунку

У ході аналізу ринку та цільової аудиторії було сформульовано список першочергових вимог, які мають бути реалізовані в системі під час розробки.

Вимоги до застосунку були розбиті на наступні групи:

- Реєстрація користувача та профіль;
- Підтвердження власності NFT;
- Маркетплейс бренду NFT;
- Процес покупки продукту;
- Історія транзакцій та відстеження замовлень;
- Керування NFT;
- Підтримка та допомога;
- Безпека та приватність;
- Співпраця бренду та випуски NFT;
- Зворотний зв'язок та відгуки;

В таблиці 2.1 наведено список функціональних можливостей (user stories) для групи «Реєстрація користувача та профіль».

Таблиця 2.1 - Вимоги до групи «Реєстрація користувача та профіль»

Назва функціоналу	Опис
Як користувач, я хочу створити обліковий запис за допомогою своєї електронної пошти або облікових записів у соціальних мережах.	Компоненти системи: Інтерфейс реєстрації, система автентифікації, база даних користувачів. Мета: Забезпечення користувачам можливості створення облікового запису за допомогою їхніх існуючих електронних адрес або облікових записів у соціальних мережах.
Як користувач, я хочу налаштувати та підключити свій гаманець MetaMask до платформи.	Компоненти системи: Інтерфейс налаштування гаманця, блокчейн мережа, збереження налаштувань користувача. Мета: Забезпечення можливості користувачам використовувати свої гаманці MetaMask для взаємодії з платформою.
Як користувач, я хочу створювати та керувати своїм профілем, включаючи додавання особистих даних та адрес гаманців.	Компоненти системи: Особистий кабінет користувача, модуль управління профілем, інтерфейс внесення особистих даних, система збереження та оновлення даних користувача. Мета: Надання користувачам можливості додавати та оновлювати свої особисті дані, включаючи адреси гаманців для використання на платформі.

В таблиці 2.2 наведено список функціональних можливостей (user stories) для групи «Підтвердження власності NFT».

Таблиця 2.2 - Вимоги до групи «Підтвердження власності NFT»

Назва функціоналу	Опис
Як користувач, я хочу, щоб платформа перевіряла мою власність певних NFT, пов'язаних з брендами.	Компоненти системи: Модуль верифікації NFT, Блокчейн мережа, База даних NFT, Інтерфейс користувача. Ця функція вимагає наявності модуля верифікації NFT, що взаємодіє з блокчейн мережею та базою даних NFT. Коли користувач звертається до платформи зі зверненням про перевірку власності певних NFT, система перевіряє цю інформацію на блокчейні та забезпечує відповідну відповідь на інтерфейсі користувача.
Як користувач, я хочу бачити свої власні NFT від підтримуваних брендів у своєму профілі.	Компоненти системи: Модуль управління профілем, Інтерфейс внесення даних, База даних NFT, Особистий кабінет користувача. Ця функція потребує модуля управління профілем та взаємодії з базою даних NFT. Коли користувач хоче додати або відобразити свої NFT від підтримуваних брендів у своєму профілі, система взаємодіє з базою даних NFT та відображає власні NFT на особистому кабінеті користувача для показу іншим користувачам або для власних потреб.

В таблиці 2.3 наведено список функціональних можливостей (user stories) для групи «Маркетплейс бренду NFT».

Таблиця 2.3 - Вимоги до групи «Маркетплейс бренду NFT»

Назва функціоналу	Опис
Як користувач, я хочу переглядати ринок, на якому відображаються преміальні товари, пов'язані з різними брендами.	Компоненти системи: Графічний інтерфейс користувача (UI), база даних товарів та їх параметрів. Мета: Дозволити користувачам оглядати широкий асортимент преміальних товарів, пов'язаних із різними брендами, на одному ринку.
Як користувач, я хочу фільтрувати продукти за брендами, для яких я володію відповідними NFT.	Компоненти системи: Механізм фільтрації товарів за певними категоріями брендів, база даних NFT власності користувачів. Мета: Надати можливість користувачам відфільтрувати товари за брендами, для яких вони володіють відповідними NFT, сприяючи більш ефективному пошуку і покупцеві.
Як користувач, я хочу переглядати детальну інформацію про продукти, включаючи зображення, описи та ціни в Ethereum.	Компоненти системи: Модуль відображення деталей товарів (фотографії, описи, ціни в Ethereum), інтерфейс відображення даних. Мета: Надати користувачам повну інформацію про товари, щоб допомогти їм прийняти обгрунтоване рішення про покупку.

В таблиці 2.4 наведено список функціональних можливостей (user stories) для групи «Процес покупки продукту».

Таблиця 2.4 - Вимоги до групи «Процес покупки продукту»

Назва функціоналу	Опис
Як користувач, я хочу додати бажані товари у кошик та перейти до оформлення замовлення.	Компоненти системи: Корзина покупок, інтерфейс вибору товарів, модуль підтвердження замовлення. Мета: Надати можливість користувачам вибирати бажані товари та зберігати їх у кошику для подальшого замовлення, спрощуючи процес покупки.
Як користувач, я хочу здійснити покупку за допомогою Ethereum з свого гаманця MetaMask.	Компоненти системи: Інтеграція з гаманцем MetaMask, модуль оплати, блокчейн мережа Ethereum. Мета: Забезпечити можливість користувачам проводити покупки з використанням Ethereum, використовуючи їхні гаманці MetaMask, що забезпечує швидкість та безпеку операцій.
Як користувач, я хочу, щоб система перевіряла моє володіння необхідним брендом NFT перед завершенням покупки.	Компоненти системи: Модуль перевірки NFT, база даних NFT власності користувачів. Мета: Перевірити, що користувачі фактично володіють необхідним NFT бренду перед тим, як здійснити покупку, забезпечуючи безпеку та конфіденційність операцій.

В таблиці 2.5 наведено список функціональних можливостей (user stories) для групи «Історія транзакцій та відстеження замовлень».

Таблиця 2.5 - Вимоги до групи «Історія транзакцій та відстеження замовлень»

Назва функціоналу	Опис
Як користувач, я хочу переглядати історію своїх транзакцій, включаючи минулі покупки та їх деталі.	Компоненти системи: Особистий кабінет користувача, модуль історії транзакцій, база даних покупок. Мета: Надати користувачам зручний доступ до інформації про їхні минулі покупки, дозволяючи їм відстежувати та аналізувати свої попередні операції на платформі.
Як користувач, я хочу відстежувати статус своїх замовлень, включаючи інформацію про доставку та оновлення щодо доставки, якщо це можливо.	Компоненти системи: Модуль відстеження замовлень, інтеграція з системою доставки, база даних замовлень. Мета: Надати користувачам можливість відстежувати процес їхніх замовлень, включаючи інформацію про стан доставки, що забезпечує прозорість та комфорт в процесі отримання замовлених товарів.

В таблиці 2.6 наведено список функціональних можливостей (user stories) для групи «Керування NFT».

Таблиця 2.6 - Вимоги до групи «Керування NFT»

Назва функціоналу	Опис
Як користувач, я хочу мати можливість передавати чи дарувати власні NFT конкретних брендів іншим користувачам.	Компоненти системи: Модуль передачі/дарування NFT, інтерфейс користувача для вибору NFT, взаємодія з блокчейн мережею для переведення власності NFT. Мета: Надання користувачам можливості обмінюватися або дарувати власні NFT іншим користувачам, розширення функціональності та сприяння соціальній взаємодії між учасниками платформи.
Як користувач, я хочу переглядати та керувати колекцією NFT, пов'язаних із підтримуваними брендами.	Компоненти системи: Особистий кабінет користувача, модуль управління колекцією NFT, інтерфейс для перегляду та сортування NFT, взаємодія з базою даних NFT. Мета: Надання користувачам можливості зручно керувати своїми власними NFT, відображення та організація колекцій, пов'язаних з різними підтримуваними брендами, для зберігання та організації цифрових активів.

Обов’язково, щоб користувач мав доступ до необхідної інформації, тому розділ «Підтримка і допомога» повинен містити чітку та корисну інформацію щодо застосунку.

В таблиці 2.7 наведено список функціональних можливостей (user stories) для групи «Підтримка та допомога».

Таблиця 2.7 - Вимоги до групи «Підтримка та допомога»

Назва функціоналу	Опис
Як користувач, я хочу мати доступ до служби підтримки для будь-яких питань або проблем, що стосуються покупок, NFT або платформи в цілому.	Компоненти системи: Система зв'язку з користувачем (наприклад, чат або тікет-система), база знань або система керування знаннями для реєстрації, відстеження та вирішення запитів користувачів. Мета: Надання зручного способу користувачам звертатися з будь-якими питаннями чи проблемами на платформі, допомога у вирішенні проблем та забезпечення якісного обслуговування користувачів.
Як користувач, я хочу мати розділ з частими запитаннями або повністю проставленими посібниками, щоб допомогти мені орієнтуватися на платформі та її можливостях.	Компоненти системи: Сторінка частих запитань (FAQ), інтерфейс для навігації по посібникам, база знань з вичерпним описом функцій та можливостей платформи. Мета: Надання користувачам простого та зручного способу отримання інформації про платформу, її функціонал та навігацію, що допоможе їм краще розуміти та використовувати всі можливості платформи.

Як веб-застосунок, що надає можливість проведення транзакцій для купівлі та продажу товарів, повинен надати гарантію безпеки даних користувача.

В таблиці 2.8 наведено список функціональних можливостей (user stories) для групи «Безпека та приватність».

Таблиця 2.8 - Вимоги до групи «Безпека та приватність»

Назва функціоналу	Опис
Як користувач, я хочу мати гарантію, що мої особисті дані, деталі транзакцій та інформація про гаманець захищені.	Компоненти системи: Система шифрування даних, захищений протокол передачі даних, аутентифікаційні системи, захист гаманця користувача. Мета: Забезпечення високого рівня захисту особистих даних користувачів та їхніх транзакцій, що стає важливим для платформи, що пропонує обмін цифровими активами та забезпечує конфіденційність і безпеку користувачів.
Як користувач, я хочу керувати своїми налаштуваннями конфіденційності та контролювати, яка інформація є видимою у моєму профілі.	Компоненти системи: Панель управління конфіденційністю, інтерфейс налаштувань приватності, система контролю доступу до особистої інформації користувача. Мета: Надання користувачам можливості керувати своєю особистою інформацією та контролювати, яка інформація є доступною для інших користувачів, забезпечуючи більш високий рівень приватності і відчуття контролю над їхніми персональними даними на платформі.

В таблиці 2.9 наведено список функціональних можливостей (user stories) для групи «Співпраця бренду та випуски NFT».

Таблиця 2.9 - Вимоги до групи «Співпраця бренду та випуски NFT»

Назва функціоналу	Опис
Як користувач, я хочу отримувати сповіщення про нові випуски NFT від брендів, які я відстежую або володію NFT.	Компоненти системи: Система сповіщень, модуль підписки на оновлення, інтерфейс відстеження нових випусків. Мета: Надати користувачам можливість бути в курсі нових випусків NFT від брендів, які вони цікавляться, що сприяє підвищенню залученості та інтересу до платформи.
Як користувач, я хочу брати участь у спеціальних співпрацях брендів або обмежених випусках NFT.	Компоненти системи: Модуль участі в співпраці, система обмежених випусків NFT, інтерфейс реєстрації на участь в спеціальних випусках. Мета: Дозволити користувачам брати участь у спеціальних акціях та обмежених випусках NFT, що створює додатковий інтерес та можливість отримання унікальних активів.

При появі запитань, які описують нестандартні ситуації, необхідно, щоб користувач мав можливості зв’яку зі службою підтрики, яка швидко надасть будь-які відповіді стосовно користування системою.

В таблиці 2.10 наведено список функціональних можливостей (user stories) для групи «Зворотний зв'язок та відгуки».

Таблиця 2.10 - Вимоги до групи «Зворотний зв'язок та відгуки»

Назва функціоналу	Опис
Як користувач, я хочу залишати відгуки та оцінки для продуктів, які я придбав.	Компоненти системи: Модуль відгуків та оцінок, інтерфейс залишення відгуків. Мета: Дозволити користувачам висловлювати свою думку про придбані продукти, що надає іншим учасникам можливість оцінити їх якість та корисність.
Як користувач, я хочу переглядати відгуки та оцінки від інших користувачів для прийняття обґрунтованих рішень щодо покупок.	Компоненти системи: Інтерфейс перегляду відгуків та оцінок, фільтри для сортування відгуків. Мета: Допомогти користувачам приймати обґрунтовані рішення при покупці, адже вони можуть переглядати думки інших користувачів щодо продуктів.

2.3 Основні компоненти блокчейн-застосунків

2.3.1 Характеристики блокчейну та типи технології

Блокчейн визначається як технологія розподіленого реєстру, яка управляється різними вузлами на мережі взаємодії "peer-to-peer" [19]. Ця система функціонує без централізованого управління зберіганням даних чи центральних адміністраторів. Дані зазвичай розподіляються по різних вузлах, а реплікація та шифрування забезпечують цілісність даних. Ідея блокчейну виникла 31 жовтня 2008 року в статті, написаній Сатоші Накамотою. Він придумав концепцію транзакцій Bitcoin у мережі, де онлайн-платежі можна відправляти безпосередньо від одного учасника до іншого, обходячи банківські установи. Основна ідея Сатоші була розробити "ланцюг блоків", який вирішує проблему подвійного витрачання шляхом використання технології розподіленого реєстру "peer-to-peer" та обчислення

хронологічного порядку цифрових транзакцій. Коли ми говоримо про блокчейн, ми маємо на увазі ланцюжок, який містить блоки, а кожен блок у ланцюжку зберігає інформацію про всі транзакції, які відбуваються у будь-який момент часу. Отже, кожен блок відіграє важливу роль у зв'язку з заголовком блоку та наступним блоком, щойно частина ланцюжка потрапляє до системи.

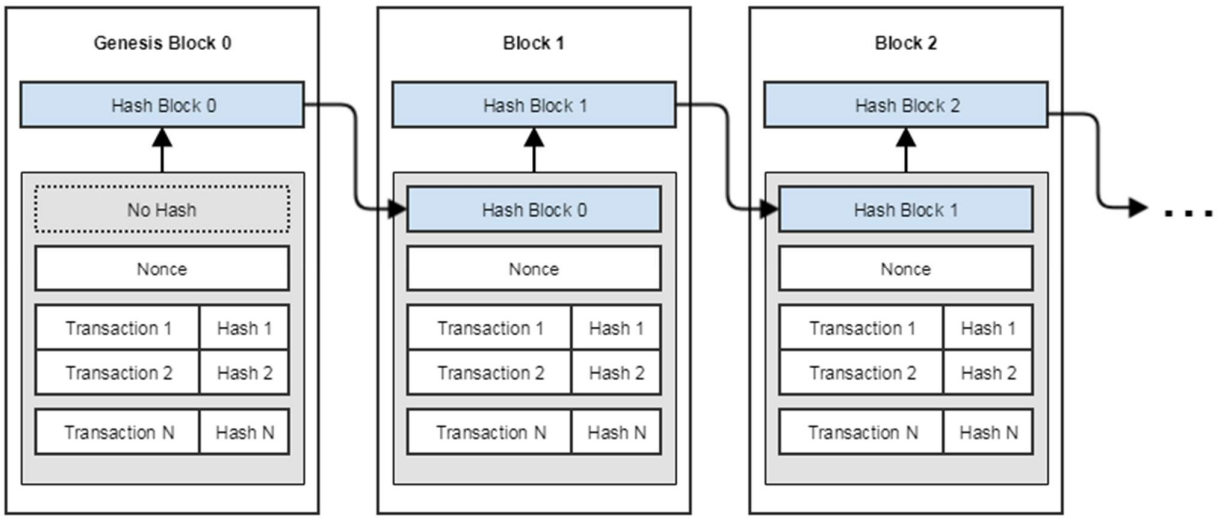


Рисунок 2.2 – Схема зв’язків ланцюгів блокчейну

Кожен блок має основну функцію запису, підтвердження та розподілу транзакцій серед інших блоків. Це означає, що блок у системі не може бути виключений чи значно змінений, оскільки зміниться кожний наступний блок. Тому система є розподіленою інформаційною системою, яка містить інформацію про всі минулі транзакції і продовжує працювати за попередньо визначеним протоколом, що визначає напрямок здійснення та підтвердження транзакцій, а також функціональність всієї системи та її учасників. Крім того, ця мережа зазвичай відноситься до децентралізованого реєстру, оскільки інформація зберігається на кожному вузлі, який має дозвіл на операції у кожному з каналів.

Група транзакцій у мережах блокчейну об'єднана у ланцюжки, з'єднані за допомогою блоків, використовуючи хеші попередніх блоків у реєстрації [20]. Та-

ким чином, основна властивість безпеки мереж блокчейну забезпечується як властивість незмінності. На рисунку 2.3 зображено схему виявлення транзакцій зловмисника.

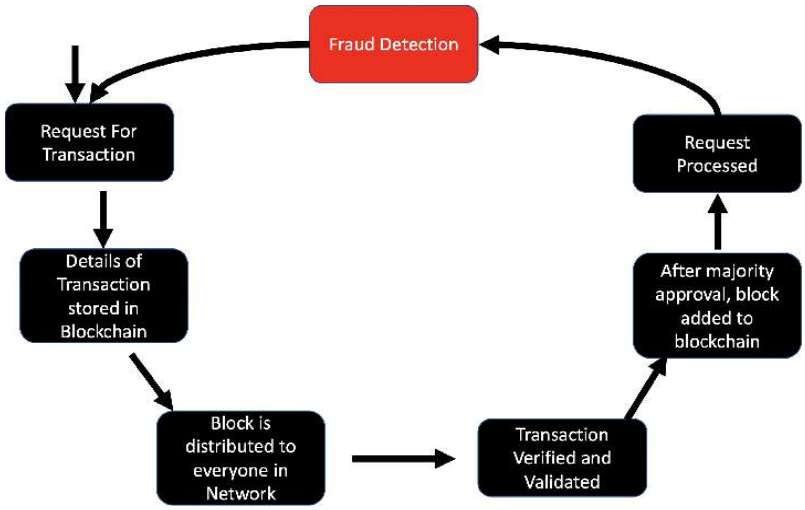


Рисунок 2.3 – Виявлення транзакції зловмисника

Чим далі блок у ланцюжку (чим пізніше він), тим безпечнішою від змін є інформація, що міститься в ньому. Коли зловмисник намагається змінити деякі ключі, місцевий запис втрачає чинність негайно, оскільки значення хешів у заголовках наступних блоків змінюються повністю на основі криптографічної хеш-функції [21].

В основному, існує три основних типи блокчейнів: публічні(public), приватні(private) та консорціумні(consortium) блокчейни [22]. Ці типи технологій відрізняються за використанням, управлінням дозволами та характеристиками роботи, що дозволяє кожному типу блокчейну застосовуватися у різноманітних додатках. Найважливіші характеристики цих типів блокчейнів порівняно та обговорено у Таблиці 2.11.

Таблиця 2.11 - Порівняння типів блокчейнів та характеристик

	Public Blockchain	Private Blockchain	Consortium Blockchain
База даних	Кожен учасник може отримати доступ до бази даних, зберегти копію транзакції та змінити їх.	Центральний орган, який керує правами доступу до бази даних або можливістю змін.	Відкритий для громадськості, але не всю інформацію можна доступно отримати для них.
Рівень безпеки	Дуже безпечний, оскільки кожен блок має копію окремої транзакції.	Деякі особи мають доступ, і це вважається безпечним, оскільки всі учасники відомі.	Забезпечує конфіденційність транзакцій.
Вартість	Висока вартість	Нижча за публічні блокчейни	Нижча за публічні блокчейни
Швидкість	Повільна швидкість, оскільки кожну транзакцію потрібно перевіряти та синхронізувати з кожним вузлом.	Висока швидкість через обмежену кількість учасників.	Висока швидкість порівняно з публічним блокчейном.
Протокол консенсусу	Кожен може брати участь у процесі консенсусу.	Механізм консенсусу підтримує велику кількість учасників.	Процес консенсусу контролюється через попередньо вибрані вузли.
Розширення блоків	Розмір блоку продовжує збільшуватиісь.	Залучено лише кількох учасників.	Розмір блоку контролюється, оскільки участь беруть лише відомі учасники.

Група транзакцій у мережах блокчейну об'єднана у ланцюжки, з'єднані за допомогою блоків, використовуючи хеші попередніх блоків у реєстрації [23]. Таким чином, основна властивість безпеки мереж блокчейну забезпечується як властивість незмінності. Чим далі блок у ланцюжку (тим пізніше він), тим безпечнішою від змін є інформація, що міститься в ньому. Коли атакувач намагається змінити деякі ключі, місцевий запис втрачає чинність негайно, оскільки значення

хешів у заголовках наступних блоків змінюються повністю на основі криптографічної хеш-функції.

2.3.2 Смарт-контракти на основі блокчейну

Смарт-контракти є невід'ємною складовою блокчейн-орієнтованих застосунків. Це угода, укладена між різними сторонами, що беруть участь у визначеній системі. Це комп'ютерний протокол, який відповідає конкретним правилам, кодам та обмеженням, на які погодилися всі учасники в мережі. Наприклад, розумна банківська транзакція або контракт фінансової спрямованості включає всі умови, до яких погодилися кожен зацікавлений учасник у цьому процесі [24].

Смарт-контракти, у контексті блокчейну, є виключно логікою, яка існує на блокчейні, може приймати або виконувати транзакції, так само як будь-яка адреса (транзакції можуть бути відхилені або для їх виконання може знадобитися специфічні аргументи) і може продовжувати діяти як незмінна угода [25]. Смарт-контракти призначені служити як протокол виконання умов контракту за допомогою комп'ютерів, вперше вжитий криптографом Ніком Шабо. Центральна концепція та походження елемента "контракт" у назві полягає в тому, що деякі аспекти контрактів мають бути вбудовані у код таким чином, що їх порушення буде або коштовним, або складним.

2.3.3 Блокчейн Ethereum

Ефіріум вважається перспективною та відносно новою та високоінноваційною платформою, яка з'явилася в 2015 році та має можливість створювати розподілені додатки з повністю функціональною мовою програмування, яка працює в децентралізованій, peer-to-peer мережі, такій як блокчейн [26]. Відповідно, це поле здобуло значну увагу академічної громадськості протягом останніх двох років. Ethereum - це розподілена обчислювальна платформа на основі публічного блокчейну, яка надає можливості для створення смарт-контрактів [27].

Вона забезпечує віртуальний децентралізований комп'ютер, відомий як Ethereum Virtual Machine (EVM), як рамкове середовище для виконання смарт-контрактів. Мережа Ethereum - це особлива блокчейн-мережа та платформа для використання віртуальної машини (Ethereum Virtual Machine, EVM) для запуску смарт-контрактів. Оскільки світ функціонує тільки на блокчейні у вигляді віртуальної машини, смарт-контракти на вузлах машин повністю ізольовані від мережі, файлової системи чи інших процесів.

Для написання смарт-контрактів з Ethereum була розроблена мова програмування високого рівня, яка є тьюрінг-повною. Ethereum постійно розвиває цифрові технології і значно розширює свої можливості. Це ціла мережа з власним веб-браузером, мовою програмування та торговою платформою. Найважливше, вона допомагає користувачам будувати розподілені додатки на блокчейні Ethereum. Ethereum blockchain є зручним у використанні при створенні DApps.

2.3.4 Solidity

У блокчейні Ethereum смарт-контракти зазвичай написані мовами вищого рівня, а потім перетворюються у байткод на Ethereum Virtual Machine (EVM). Мови вищого рівня включають Serpent, що має багато спільного з Python, LLL, Viper (мова, схожа на Python) та Solidity, яка дуже схожа на Javascript. Solidity є найвизначнішою та широко використовуваною мовою для розробки смарт-контрактів.

При створенні контрактів за допомогою Solidity вони мають схожість з класами в об'єктно-орієнтованих мовах програмування. Так само, як у звичайному імперативному програмуванні, код контракту складається зі змінних та функцій, які інтерпретують та змінюють їх.

Мова Solidity має кілька відомих особливостей та список майбутніх змін, що означає, що програмне забезпечення, що зараз пишеться, може не повністю

працювати з майбутнім оновленням. Є кілька найкращих практик програмування, пов'язаних з дизайном смарт-контрактів, які були накопичені за (відносно) короткий час, коли використовувалася мова Solidity.

Для розуміння базових структур та механізмів програми на Solidity, наведено приклад коду. Першою змінною, яку ми повинні оголосити, є адреса у контракті. Вона має значення 160-біт та зберігає адресу смарт-контракту. Публічний доступ до цієї адреси дозволить використовувати її іншим оголошеним контрактам. Мапування (mapping) встановлює зв'язок між змінними uint (цілі числа беззнакового типу) адреси.

```
1  contract cryptoCoin {
2      address public minter;
3      mapping (address => uint) public balances;
```

Рисунок 2.4 – ініціалізація смарт-контракту

Наступним кроком є оголошення події (event), яке активується при виконанні функції send (рис. 2.5). Клієнти Ethereum можуть відстежувати та контролювати транзакції, слухаючи ці події та отримуючи аргументи.

```
1  event Sent(address from, address to, uint amount);
```

Рисунок 2.5 – декларація події в смарт-контракті

Як показано на наступному малюнку, після створення контракту виконується конструктор.

```
1  constructor() public {
2          minter = msg.sender;
3      }
```

Рисунок 2.6 – Створення конструктору події

Цей контракт має дві функції - mint та send. Тільки contract creator може викликати mint і створювати вказану кількість монет, що можуть бути відправлені на іншу адресу. Функція enable визначає умови, за яких потрібно повернути внесені зміни. Наприклад, вона перевіряє, що minter дійсно є початковим контрактом, та встановлює максимальну кількість монет, які можна відправити.

```
function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e50);
        balances[receiver] += amount;
    }
```

Рисунок 2.7 – Опис функції мінтингу токену

```
function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Balance too low!");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Рисунок 2.8 – Опис функції надсилання токену

На відміну від mint, send доступний кожному, хто володіє монетами. Вони можуть відправляти токени іншій особі, успішно виконавши цю функцію. Однак, якщо відправник намагається передати більше монет, ніж він наразі володіє, то виклик потрібної функції не буде виконано, і в цьому випадку з'явиться повідомлення про помилку.

2.3.5 Metamask

MetaMask є відомим інструментом у світі криптовалют та блокчейну. Це розширення для браузера, що дає можливість користувачам безпечно взаємодіяти з децентралізованими додатками (DApps) на блокчейні.

На рисунку 2.9 зображено користувацький інтерфейс криптогаманця Metamask.

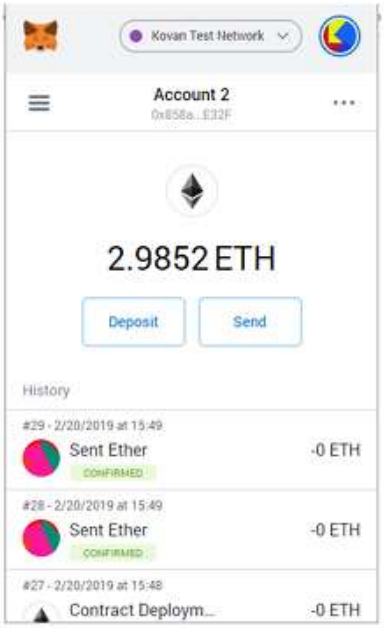


Рисунок 2.9 – користувацький інтерфейс застосунку Metamask

Однією з ключових переваг MetaMask є його можливість взаємодії з різними блокчейнами та простота у використанні. Він пропонує користувачам відмінну універсальність, спрощуючи відправку та отримання криптовалюти на різних мережах блокчейну.

Іншою ключовою рисою є те, що MetaMask дозволяє користувачам зберігати свої криптовалютні активи та керувати ними безпосередньо у своєму браузері. Він надає безпечний спосіб здійснення транзакцій та виконання операцій з криптовалютою без необхідності використання окремого гаманця.

2.4 Обгрунтування вибору програмного забезпечення

У виборі бази даних для дипломного проекту було досліджено численні варіанти, серед яких MySQL, Oracle та Microsoft SQL Server. Але після докладного порівняння, було вирішено, що PostgreSQL найкраще відповідає потребам та цілям роботи.

PostgreSQL є відкритим програмним забезпеченням, що має активну спільноту. Це означає, що можна розраховувати на надійну підтримку, своєчасні оновлення та покращення безпеки. Хоча і інші бази даних також популярні, PostgreSQL відрізняється своєю гнучкістю та відгуками на спільноту[28].

Щодо стабільності, PostgreSQL славиться своєю надійністю та довговічністю. Це важливо, оскільки є вимога, щоб даний проект працював безперебійно.

Крім того, PostgreSQL може легко розширюватися, що ідеально підходить для потреб роботи, адже планується розширення проекту у майбутньому. Ця база даних також відповідає стандартам SQL, що спрощує перенесення даних.

Однією з ключових особливостей PostgreSQL є його функціональність. Він пропонує широкий спектр можливостей, таких як обробка геоданих, текстовий пошук і підтримка різних розширень.

Останнім, але не менш важливим, є питання вартості. PostgreSQL є безкоштовним, тоді як деякі інші бази даних можуть вимагати комерційної ліцензії, що збільшує витрати.

Таким чином, обравши PostgreSQL, надається забезпечення проекту високоякісною, надійною та економічно вигідною базою даних.

При розробці додатків, що спрямовані на використання блокчейну, вибір відповідної мови програмування та відповідного середовища є критично важливим. Це визначає ефективність, можливість масштабування та співпрацю з блокчейн-технологіями. Один з варіантів — це TypeScript. Однією з ключових особливостей TypeScript є його статична типізація. Вона дає змогу розробникам виявляти та виправляти помилки ще до фази виконання програми. Такий підхід забезпечує більшу стабільність коду, що є особливо актуальним при роботі з такими складними структурами, як блокчейн [29].

Крім того, TypeScript пропонує покращені можливості для об'єктно-орієнтованого програмування. Завдяки цьому, розробники можуть створювати код, який не тільки легко розширюється, але й легко підтримується в довгостроковій перспективі.

У контексті розробки на основі блокчейну, Node.js виявляється чудовим вибором завдяки декільком ключовим особливостям. Однією з них є асинхронність, яка дозволяє ефективно працювати з блокчейн-мережами, де швидкість відгуку може бути вирішальною. Крім того, завдяки легкості масштабування Node.js, розробники можуть легко розширювати та вдосконалювати свої додатки, особливо коли йдеться про великі проекти, що інтенсивно взаємодіють з блокчейн-системами.

Що стосується інтеграції з блокчейн, існує безліч готових рішень для TypeScript та Node.js, включаючи бібліотеки, які спрощують взаємодію з блокчейн-мережами, такими як Ethereum. Для цієї платформи існує чимало npm-пакетів, що полегшують роботу з розумними контрактами.

Серед інших мов програмування, таких як Java та C#, можна відзначити, що вони можуть вимагати більше коду та бути менш гнучкими в плані розробки, особливо при інтеграції з блокчейном. Хоча Python також підтримує розробку на основі блокчейну, йому може не вистачати строгості типізації, а його ефективність може бути нижчою в розподілених системах.

Загалом, вибір Node.js та TypeScript для розробки блокчейн-орієнтованих додатків забезпечує розробникам потрібні інструменти для швидкої та надійної розробки, а також спрощує процес інтеграції з різними блокчейн-технологіями.

При розробці фронтенду додатка ключовим є вибір відповідного фреймворку, адже він безпосередньо впливає на якість користувацького інтерфейсу, швидкість створення продукту та можливість додавання нових функцій у майбутньому. З усіх доступних на ринку варіантів було вирішено зосередитись на Vue.js. Одна з причин – це тому, що Vue.js відомий своєю інтуїтивністю та легкістю в освоєнні, що спрощує процес адаптації розробників до нового інструменту. У порівнянні з іншими фреймворками, такими як React або Angular, Vue.js виглядає менш заплутаним, особливо для тих, хто тільки розпочинає свій шлях у фронтенд-розробці.

Однією з ключових особливостей Vue.js є його реактивність. Ця парадигма робить процес відстеження та оновлення стану компонентів набагато прозорішим і ефективнішим, що забезпечує більш гладке та динамічне взаємодію користувача з додатком.

Вибір на користь Vue.js базується на балансі його простоти, продуктивності та зручності в роботі, що робить його ідеальним інструментом для розробки додатків, які планують взаємодію з блокчейн технологіями [30].

При створенні децентралізованих додатків (DApps) та смарт-контрактів на блокчейні основоположним є вибір правильної мови програмування. Було обрано мову Solidity, яка призначена для платформи Ethereum.

Solidity була створена з метою спрощення розробки смарт-контрактів для Ethereum. Її інструментарій ідеально підходить для взаємодії з Ethereum Virtual Machine (EVM), що робить її виділяючою серед інших мов, таких як Vyper чи Michelson, які, хоча й підтримуються, проте не настільки добре пристосовані для Ethereum, як Solidity.

Однією з найсильніших сторін Solidity є її підтримка широкого спектра можливостей. Ця мова надає інтеграцію з такими токенами, як ERC-20, ERC-721 та ERC-1155, що спрощує процес створення та обробки Ethereum-базованих токенів. У порівнянні з іншими мовами, такими як Chaincode або Clarity, Solidity пропонує набагато більше функцій.

Solidity також відрізняється активною спільнотою розробників і відмінною документацією. Це робить її досить простою для вивчення і використання, особливо порівняно з менш популярними мовами, такими як Rust для Move або SmartPy для Michelson.

Додатково, Solidity вже стала стандартом у світі розробки смарт-контрактів для Ethereum. Це забезпечує її широке застосування у сферах децентралізованих фінансів (DeFi), NFT та інших напрямках блокчейну. В той час як інші мови, такі як Rust чи SmartPy, мають обмежені можливості застосування.

Тому, вибравши Solidity як основний інструмент, отримано мову, яка відповідає всім поставленим вимогам, зокрема з погляду підтримки Ethereum, а також забезпечує великі можливості для реалізації різноманітних функцій у децентралізованих додатках і смарт-контрактах.

2.5 Порівняння сучасних підходів до розробки та тестування веб-застосунків із запропонованим методом тестування

Сучасні підходи до розробки блокчейн-застосунків використовують різні методології, що мають на меті забезпечити високу якість, ефективність та швидкість розробки.

Проте не всі сучасні методології, враховують тонкощі комплексних систем, які використовують блокчейн технології. Як було вказано раніше, до блокчейн технологій належать:

В сучасних методологіях надають процесу тестування недостатньої уваги, якщо застосовувати їх до таких комплексних систем як блокчейн-застосунки.

Запропонований підхід використовує найкращі практики методології Agile та звертає більшу увагу саме на процеси забезпечення якості належного функціонування використовуваних компонентів блокчейн технології в застосунку. На рисунку 2.10 наведено запропонований життєвий цикл тестування програмного забезпечення.

Проте, не всі сучасні методи розробки програмного забезпечення докладно враховують особливості складних систем, які використовують блокчейн-технології.

Як було наведено вище, блокчейн-технології включають у себе наступні компоненти: блокчейн мережа, яка складається з вузлів-валідаторів, смарт-контракти, які визначають логіку поведінки децентралізованого застосунку, сканери, які зчитують інформацію зі смарт-контрактів і взаємодіють із застосунком, кри-

птографічні адреси або криптогаманці, які можуть бути приєднані до смарт-контракту або до реального користувача, щоб надати йому можливість проводити транзакції в мережі. Все це – окрема екосистема, яка потребує окремої уваги.

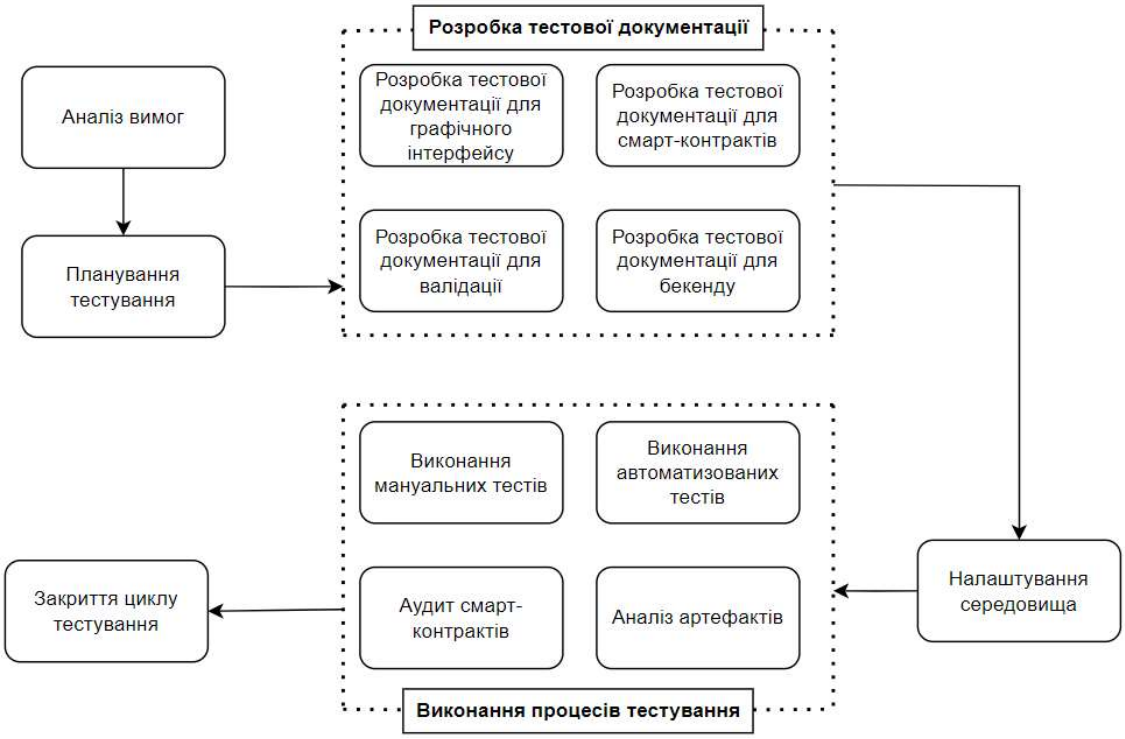


Рисунок 2.10 – життєвий цикл тестування блокчейн-застосунку (блокчейн-Agile)

Підходи до тестування блокчейн-додатків повинні бути націлені на тестування окремих складових системи, їх взаємодію та особливості функціонування.

Недоліком багатьох методологій – відсутність досить глибокого розгляду специфіки блокчейн-екосистеми, що може призвести до недоліків у тестуванні та неадекватності у функціональності або безпеці. В запропонованій методології процеси автоматизації тестування є обов’язковими, адже компоненти системи чутливі до помилок, пов’язаних з людським фактором.

Також, з урахуванням специфіки технології, різні процеси тестування повинні виконуватися спеціалістами з відповідною підготовкою та досвідом. Використовуючи інструменти тестування блокчейн застосунків, можна налагодити

безперервний процес забезпечення якості, використовуючи автоматизацію тестування та пайплайн процеси. Також це надає зручності

В таблиці 2.12 наведено порівняння різних методологій та їх процесів тестування, з наведеним підходом.

Таблиця 2.12 – Порівняльна таблиця методологій розробки та їх процесів тестування

	Waterfall	Scrum	Agile	Блокчейн-Agile
Підхід до тестування	Фазовий	В кінці кожного спринту	Під час кожної ітерації	Під час кожної ітерації
Наявність визначених процесів тестування блокчейн компонентів	X	X	X	✓
Проведення аудиту смарт-контрактів	X	X	X	✓
Гнучкість розробки	X	X	✓	✓
Враховані процеси автоматизації тестування та визначені як необхідні	X	X	X	✓

2.6 Висновки до розділу

Процес розробки блокчейн застосунку потребує попереднього планування і обґрунтованого вибору стеку технологій. Серед великого доступного набору різноманітних інструментів розробки було ретельно підібрано необхідні технології для вирішення поставлених задач.

Окрім інструментів розробки, також було проведено порівняння існуючих методологій та наведено новий підхід, який задовільняє потреби в забезпеченні якості при розробці блокчейн-застосунків – блокчейн-Agile.

На етапі тестування застосунку, окрім традиційних видів тестування, було також розглянуто види тестування, які необхідно провести і взяти до уваги такі як тестування смарт-контракту, тестування блокчейн-мережі, тестування ініціалізації токенів (minting) та проведення аудиту смарт-контрактів для забезпечення захищеності транзакцій користувачів.

3 РОЗРОБКА ІНТЕРФЕЙСУ ТА ПРОЦЕСИ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ВЕБ-ЗАСТОСУНКУ

3.1 Розробка інтерфейсу

3.1.1. Підхід до розробки інтерфейсу

Програмні інтерфейси є головним елементом будь-якої програми. Вони є мостом між користувачем і функціональністю програмного забезпечення. Добре розроблений інтерфейс забезпечує успіх програмного додатку на 80%. Гарна взаємодія з користувачем (UX) має вирішальне значення для успіху будь-якого програмного продукту, а інтерфейс відіграє ключову роль у створенні позитивної взаємодії з користувачем [32].

Існують певні критерії щодо розробки програмних інтерфейсів, які забезпечують найкращу взаємодію з користувачем.

Одним із найважливіших факторів, який слід враховувати при розробці програмних інтерфейсів, є узгодженість. Послідовні інтерфейси допомагають користувачам швидко навчитися користуватися програмним забезпеченням. Уніфікований інтерфейс забезпечує однорідний досвід для користувачів, незалежно від завдання, яке вони виконують. Це включає узгодженість у макеті меню, кнопок і навігації. Слід підтримувати узгодженість у всьому програмному забезпеченні, включаючи колірну схему та розмір шрифту. В інакшому випадку, часта зміна форм чи кольорової гами збиває користувача з інтуїтивного напрямку, який повинен вироблятися в ході взаємодії із застосунком.

Наступний ключовий критерій при розробці програмних інтерфейсів - це простота. Складні інтерфейси можуть заплутувати користувачів і перешкоджати їм використовувати програмне забезпечення. Інтерфейс має бути інтуїтивно зрозумілим, простим у використанні та зрозумілим. Необхідно виділити основні функції, потрібні користувачам і уникати нагромодження функціоналу, який кори-

стувачу може навіть не знадобитися. Інтерфейс має бути розроблений таким чином, щоб користувач міг керувати програмним забезпеченням, полегшуючи виконання завдань.

Зворотній зв’язок має вирішальне значення для гарної взаємодії з користувачем. Зворотний зв’язок має надаватися в режимі реального часу, щоб користувачі знали, що відбувається, коли вони виконують дію. Наприклад, під час виконання завдання має відображатися індикатор виконання, а якщо щось піде не так, — повідомлення про помилку. Зворотній зв’язок має бути наданий у чіткій та стислій формі. Форми зворотнього зв’язку - вібрація, звук сповіщення, зміна вікон чи елементів на екрані та ін.

Користувачі мають різні потреби та вподобання, і інтерфейс має бути налаштованим, щоб відповідати цим потребам. Наприклад, користувачі повинні мати можливість регулювати розмір шрифту та колірну схему відповідно до своїх уподобань. Доступ до налаштувань має бути простим і не потребуватиме передових технічних навичок. Проте також важливо зберігати узгодженість в дизайні в конфігураціях, які задає користувач.

Доступність є критично важливим фактором при розробці програмних інтерфейсів. Інтерфейс має бути розроблений для користувачів з обмеженими можливостями, зокрема з вадами зору та слуху. Це включає такі функції, як перетворення тексту в мовлення та комбінації клавіш. Інтерфейс також має бути сумісним із програмами зчитування з екрану.

Інтерфейс має забезпечувати легкий доступ до документації. Це включає навчальні посібники, онлайн-довідку та посібники користувача. Довідка та документація повинні бути написані простою мовою, щоб користувачі могли легко зрозуміти інструкції.

Нарешті, тестування має вирішальне значення для забезпечення хорошої взаємодії з користувачем. Інтерфейс слід ретельно перевірити, щоб переконатися, що він простий у використанні та забезпечує кращий досвід користувача. Це включає тестування інтерфейсу реальними користувачами для виявлення

будь-яких проблем із зручністю використання. Тестування має проводитися протягом усього процесу розробки, щоб переконатися, що будь-які проблеми виявлені та вирішені на ранній стадії [33].

3.1.2. Опис технологій, використаних в розробці інтерфейсу

У сучасному світі веб-розробки реалізація стильних та динамічних інтерфейсів для користувацьких додатків стала домінуючим фактором. За останні десятиліття на ринку з'явилося безліч інструментів та технологій, що спрямовані на забезпечення швидкості, зручності та ефективності у розробці. Одним із перспективних фреймворків для створення інтерфейсів веб-додатків, який заслуговує уваги, є Vue.js.

Vue.js представляє собою прогресивний JavaScript-фреймворк, спрямований на створення користувацьких інтерфейсів, що легко розуміються. Його основні переваги полягають у простому API, реактивності та компонентному підході. Простий API робить розробку інтерфейсів більш зручною для розробників будь-якого рівня, дозволяючи швидко створювати складні веб-додатки. Vue.js забезпечує реактивність, що автоматично оновлює відображення відповідно до змін у даних, спрощуючи роботу з інтерфейсом. Компонентний підхід дозволяє розбити інтерфейс на окремі частини, що сприяє легкості розробки, тестування та підтримки коду.

Легкість вивчення та добре документований код - основні переваги Vue.js. Цей фреймворк здатний швидко пристосовуватися для нових користувачів, завдяки доступності та чіткості документації. Існує велика кількість готових компонентів, що спрощує створення інтерфейсів, а також робить його більш універсальним для застосування в різних веб-додатках. Однак, через меншу спільноту порівняно з іншими фреймворками, можуть виникати проблеми з отриманням підтримки або плагінів.

Vue.js використовується для розробки веб-додатків, що вимагають швидкого та динамічного інтерфейсу. Його компонентна структура сприяє створенню

та перевикористанню окремих частин інтерфейсу, що дозволяє швидко відтворювати та модифікувати компоненти. Vue.js використовується для розробки компонентів, реалізації різноманітних ефектів та створення інтерактивних елементів на веб-сторінках.

3.1.3 Опис головних компонентів системи

Розроблений графічний інтерфейс передбачає виконання користувачем наступні дії:

1) Підключення криптогаманця Metamask до сервісу

2) Перегляд загальної інформації про сервіс на сторінці Home

3) Перегляд списку брендів-партнерів на сторінці Brands

4) Перегляд інформації про сервіс на сторінці About

5) Перегляд інформації користувача на сторінці Dashboard (кількість придбаних NFT, адресу криптогаманця та загальну інформацію, яку надав користувач)

6) Перегляд доступних колекцій NFT, які користувач може придбати.

В додатку Б, рисунки Б.1-Б.4 зображено графічний інтерфейс застосунку.

Для цілей тестування, було розроблено сайт, на якому розміщені певні товари певного бренду, які користувач може придбати лише маючи відповідний NFT токен. Було розроблено плагін, який проводив валідацію криптогаманця Metamask користувача, який хоче придбати унікальну річ. В ході валідації перевіряється чи є в наявності саме той NFT, який необхідно для купівлі даного товару. Якщо перевірка проходить успішно, тоді користувач може додати товар в корзину і продовжити покупки на сайті. На рисунках 3.1 – 3.2 зображено вигляд плагіну та взаємодії користувача з ним.

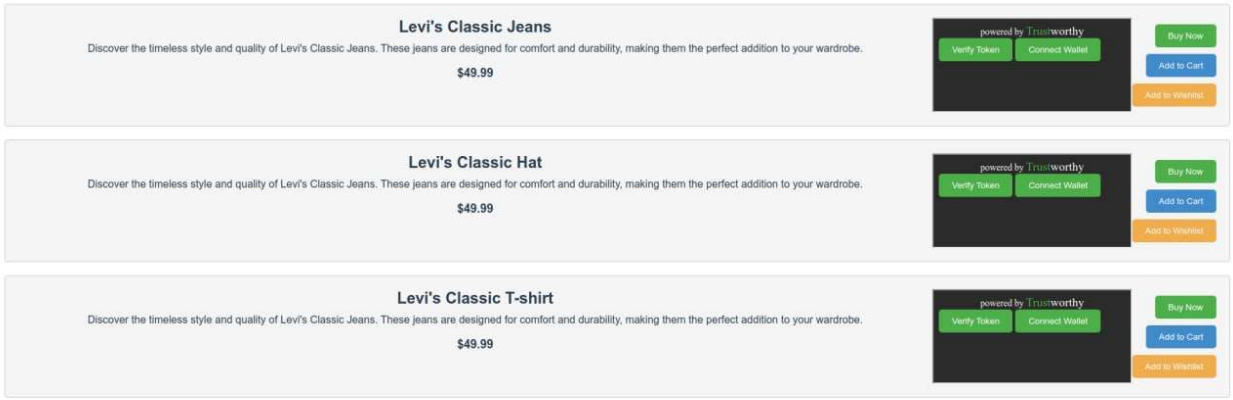


Рисунок 3.1 – Вигляд плагіну на сторінці сайту бренду

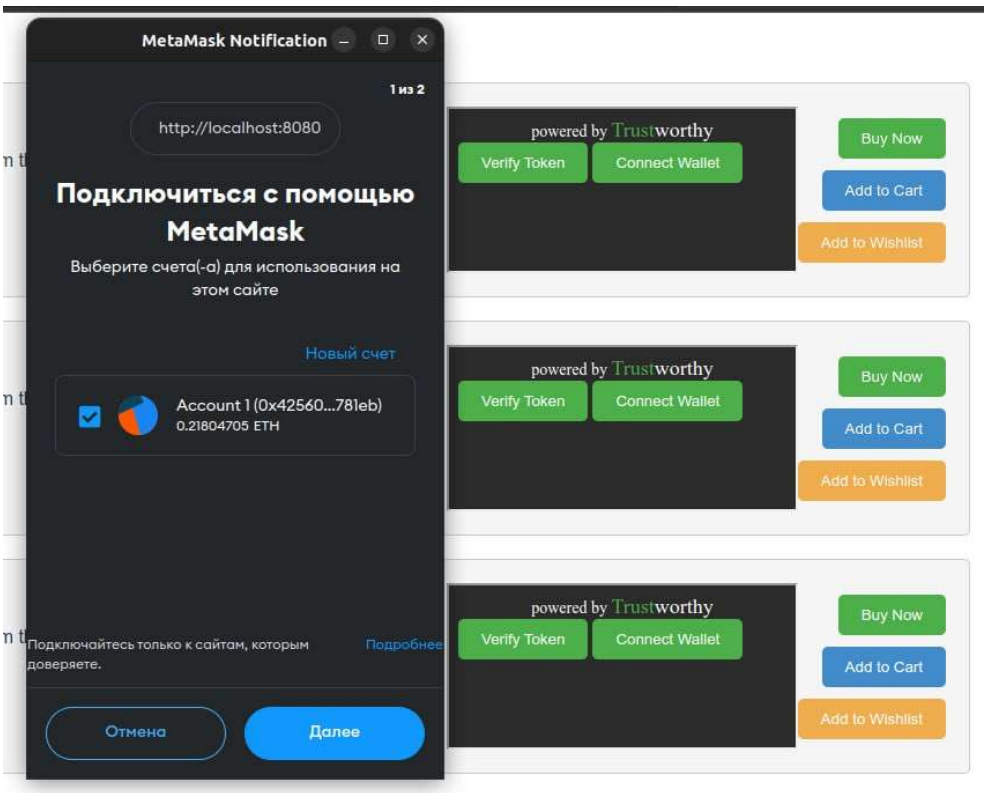


Рисунок 3.2 – Процес валідації токену для купівлі продукту

3.2 Послідовність кроків в процесі розробки програмного забезпечення

У розробці програмного забезпечення є послідовність кроків, які варто виконати перед тим, як надати користувачеві версію застосунку. Для цього використовують CI/CD (Continuous Integration and Continuous Deployment) (рис. 3.3).

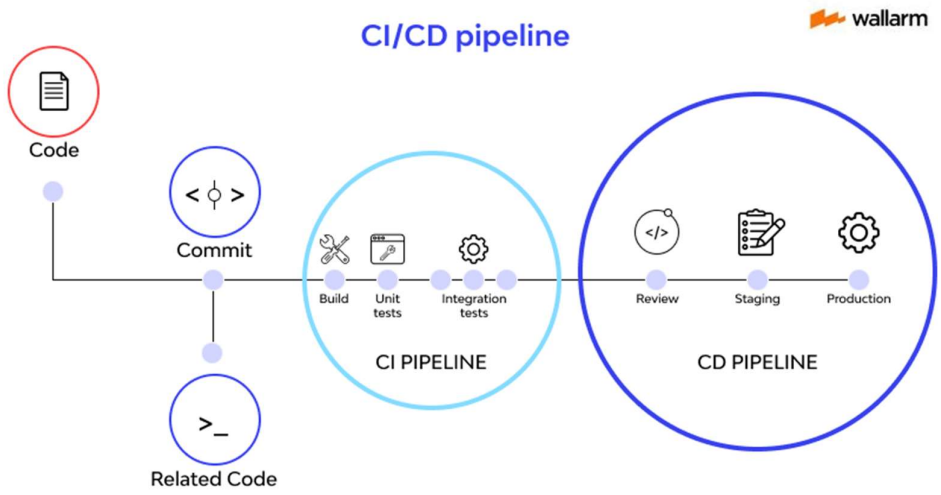


Рисунок 3.3 – Процеси Continuous Integration and Continuous Deployment

Continuous Integration (CI) та Continuous Delivery (CD) - це методи розробки програмного забезпечення, спрямовані на полегшення та автоматизацію процесу розробки, тестування та релізу програм. Continuous Integration в основному займається інтеграцією коду в спільний репозиторій розробки, тоді як Continuous Delivery описує автоматизовані процеси постійного випуску готового до релізу програмного забезпечення [34].

Continuous Integration - це практика, коли розробники регулярно інтегрують свій код в спільний репозиторій. Ця інтеграція відбувається декілька разів на день, де кожен новий код перевіряється автоматизованими тестами, а потім об'єднується з основним кодом. Це дозволяє виявляти можливі конфлікти та по-

милки з ранніх етапів розробки, зменшуючи ризик появи проблем у великих обсягах коду.

Continuous Delivery - це надалі розвиток ідеї CI. Він описує практику автоматизованої підготовки та випуску програмного забезпечення в продакшн. В процесі CD кожне нове змінення в коді проходить через автоматизовані тести, після чого готується для релізу. Це включає в себе різноманітні етапи, такі як автоматизовані тести, оцінка коду, збірка програми, автоматизоване тестування та автоматичне розгортання відповідно до заданих критеріїв.

Узагальнюючи, CI спрямований на постійне об'єднання та перевірку коду розробників, тоді як CD забезпечує автоматизацію процесу випуску програми в продакшн після успішного завершення CI. Обидва ці підходи спрямовані на підвищення швидкості, надійності та ефективності процесу розробки програмного забезпечення.

Зазвичай, попередньо визначену послідовність завдань (jobs) автоматично запускають під час впровадження змін.

До впроваджених змін можуть відноситися:

- новий функціонал, який потребує перевірки на сумісність з існуючим кодом;
- виправлення помилок, які можуть вплинути на роботу системи в цілому;
- зміна системних налаштувань;
- оновлення залежностей.

Пайплайн може включати в себе різні етапи, які можуть охоплювати перевірку нового функціоналу, виправлення помилок, оновлення конфігурацій, або ж підготовку до релізу нової версії програми. Кожен етап пайплайна має свою мету: від компіляції та тестування коду до аналізу якості коду та збереження виконуваних файлів.

В даному проєкті було розроблено описаний вище пайплайн, використовуючи інструмен Gitlab. На рисунку 3.4 зображено послідовність процесів, які повинні виконатися перед тим, як впровадити зміни на головне оточення(environment).

Рисунок 3.4 – Послідовність процесів проєкту(Pipeline)

Дана послідовність процесів забезпечує неперервну інтеграцію і автоматично виконує наступні процеси:

- build;
- unit_test;
- component_test;
- smart_contract_test;
- truffle test.
- blockchain_transaction_test;
- e2e_test;
- deploy_sandbox;

Процес «build» в цьому пайплайні виконується з метою підготовки та збирання всього блокчейн-застосунку, включаючи його API частину. Він виконує такі задачі:

- Конфігурація середовища: Установка необхідних залежностей, налаштування середовища розробки, інсталяція відповідних версій пакетів та інструментів.

- Компіляція веб-застосунку: Запуск збірки веб-застосунку для створення виконуваних файлів (HTML, CSS, JS) з вихідного коду.

- Створення та запуск API: Блокчейн-застосунок включає API частину,

яка обробляє запити, взаємодіє з блокчейном та надає доступ до функцій додатку. Підпроцес build запускає серверну частину цього API, і впевнюється, що вона працює належним чином.

- Запуск блокчейну: Під час процесу збірки проводиться ініціалізація блокчейну, включаючи його запуск, конфігурацію та інтеграцію з веб-застосунком та його API.

- Тестування функцій блокчейну та API: Це включає запуск автоматизованих тестів для перевірки функціональності та працездатності блокчейну і API частини.

- Створення документації API: Генерація документації для API, яка описує доступні ендпоінти, їх функції та параметри (swagger).

3.3 Характеристика процесів забезпечення якості веб застосунку

3.3.1 Планування тесового процесу

Тестування систем, які використовують блокчейн технології, складається з таких етапів [35]:

– ініціація;

– проектування;

– тестування;

– створення звітності проведених процесів тестування.

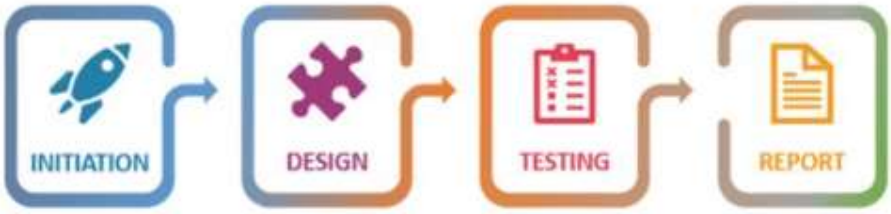


Рисунок 3.5 – Етапи тестування систем на основі блокчейн технології

Згідно із рисунком 3.5, першим етапом є ініціація процесу тестування, що передбачає усвідомлення структури блокчейну. Ця стадія включає в себе аналіз бізнес-вимог та функціональних особливостей. Тобто, детальний опис поведінки додатку та взаємодії з ним користувача. Крім того, на цьому етапі необхідно розробити повну стратегію тестування, яка включає детальне уточнення підходів до випробування програмного продукту.

Далі в послідовності йде етап проектування. Ця фаза складається з ряду процедур:

- Створення стратегії тестування або тест-плану.

- Розроблення тест-кейсів, де команда фахівців з контролю якості створює послідовність дій для кожної перевірки, бажано, щоб ці дії перевірялися бізнес-аналітиками.

- Підготовка тестових даних, що базуються на бізнес-вимогах до продукту і можуть бути створені вручну або з використанням різних інструментів.

- Налаштування тестового середовища для проведення тестування.

- Визначення метрик продуктивності задля отримання інформації стосовно ефективності системи або її складових частин.

На етапі проектування особлива увага приділяється створенню тестових документів, таких як тест-стратегія та тест-план. Тест-стратегія - це високорівневий документ, який описує рівні тестування та підходи до тестування на цих рівнях. Вона є дієвою на рівні компанії або програми (одного або кількох проектів). Тест-план - це документ, що описує ресурси, методи, графік робіт та ресурси, необхідні для проведення тестування. Крім того, він визначає інструменти тестування, функціональність, яку потрібно протестувати, розподіл ролей у команді, тестове середовище, методики тест-дизайну, критерії початку та завершення тестування, а також ризики.

Якісно підготовлений тест-план повинен включати наступну інформацію:

- об'єкт тестування - що саме підлягає перевірці,
- середовище тестування,
- додаткове обладнання та програми;

- повний перелік функціоналу, який має бути протестований;
- повний перелік методів тестування, які слід використовувати, і їх застосування до об'єкта тестування;
- перелік інструментів, які мають бути використані під час процесу тестування;
- логічно структуровані перевірки: підготовчі роботи, процес тестування та аналіз отриманої інформації щодо запланованих етапів розробки проекту.

Необхідно також відзначити, що існують два типи тест-планів: майстер-тест-план і деталізований тест-план. Майстер-тест-план вважається більш статичним, оскільки містить у своїй структурі високорівневі дані, які не піддаються постійному використанню під час здійснення контролю якості. Щодо деталізованого тест-плану, то він містить більш конкретну інформацію про стратегію оптимального тестування та повний графік виконаних робіт. Отже, цей тип плану можна вважати більш "живим" джерелом тестових маніпуляцій, яке постійно піддається редагуванню, і в ньому можна знайти більш реальний стан речей під час тестування розроблюваного продукту.

3.3.2 Види тестування систем

В розумінні звичних методів тестування виділяють такі види:

- функціональне тестування,
- нефункціональне тестування,
- тестування продуктивності,
- тестування безпеки
- інтеграційне тестування [36].

Однак, наряду з цими стандартними методами, розробники повинні мати спеціалізовані навички та можливості для ефективного тестування продуктів, що базуються на блокчейн технології. Серед них можна виділити: тестування смарт-контрактів, тестування блокчейн мереж, володіння математичними та криптографічними знаннямивикористання передових інструментів в сфері блокчейн.

Об'єднуємо загальновідомі підходи до тестування з унікальними методами перевірки систем, що використовують технологію блокчейн (рис. 3.6).

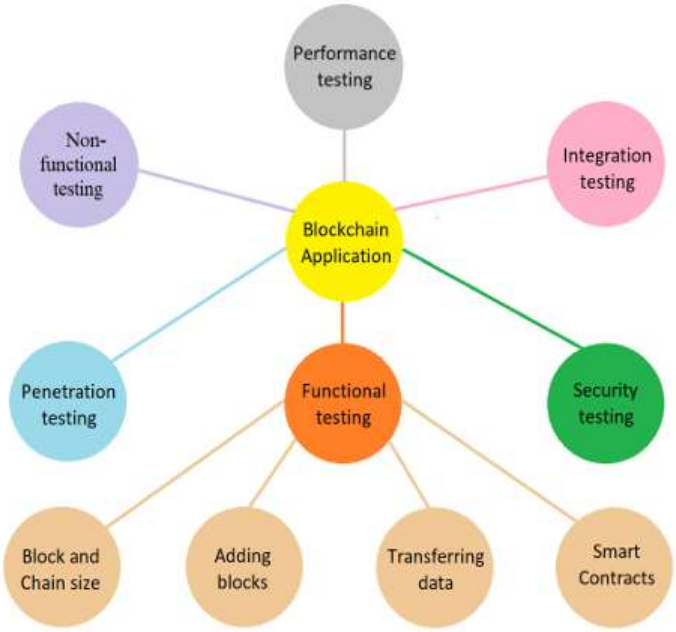


Рисунок 3.6 – Сукупність методів тестування застосунків, що використовують блокчейн

Для оцінки бізнес-вимог і ефективності сценаріїв використання функціональне тестування є критично важливим. Основні моменти функціонального тестування блокчейн-додатків [37].

– перевірка розміру блоку та ланцюга;
– перевірка можливості передачі даних;
– перевірка можливості додавання блоків;
– перевірка смарт-контрактів.

Розмір пам’яті кожного блоку блокчейну вказується в мегабайтах. З міркувань безпеки розмір пам’яті був зменшений з 36 мегабайтів до 1 мегабайта. Тестувальники повинні зосередитися на виборі даних транзакції, методах шифрування, які слід використовувати для пов’язання цих блоків, і подібних складних сценаріях. Необхідно також звернути увагу на наявність втрати даних під час

транзакцій між блоками, тому що основна архітектура ланцюжка блоків.

В ході передачі даних у мережі блокчейн важливо систематично тестувати процес відправки між блоками, оскільки надійність і безпека інформації є ключовими складовими у цій системі. Додавання нових блоків до ланцюжка вимагає пристрасної оцінки, оскільки вони стають необоротними після включення у ланцюжок. Розробка та дотримання смарт-контрактів відіграє важливу роль у забезпеченні безперебійного функціонування системи, оскільки це впливає на взаємодію між учасниками та їх відповідальність за правила угод. Також критично важливе незалежне тестування різноманітних вузлів, що існують у мережі, оскільки це забезпечить їх стабільну та безперервну працездатність в системі блокчейн.

Смарт-контракти визначають умови та правила автоматизованих угод між учасниками мережі. Ці контракти виконуються автоматично при виконанні визначених умов і не потребують додаткового підтвердження або втручання з боку третьої сторони. Для забезпечення безперебійної роботи системи блокчейн, важливо не лише створювати смарт-контракти, але й переконатися, що всі учасники мережі відповідають вимогам цих контрактів.

Процес тестування вузлів мережі включає в себе проведення різноманітних тестів для кожного вузла окремо з метою визначення їхньої працездатності та відповідності вимогам системи. Тестування вузлів включає у себе перевірку їхньої відповідності стандартам безпеки, швидкодії, надійності під час виконання різних операцій у мережі та відновлення після можливих відмов.

Цей комплексний підхід до розгляду смарт-контрактів та тестування вузлів у мережі блокчейн є необхідним для забезпечення стабільності, безпеки та ефективності функціонування системи.

У межах інтеграційного тестування систем необхідно забезпечити безперебійність взаємодії між всіма її компонентами. Блокчейн, як складна екосистема, має численні складові, що вимагають гармонійного взаємозв'язку. Тому необхідно проводити тестування різних API, що пов'язані з цими компонентами, на їхню взаємовідповідність та сумісність.

Тестування продуктивності у блокчейні концентрується на перевірці обсягів транзакцій та їх розміру, перевірці працездатності окремих блоків чи додатків, призначених для впровадження в промислове використання. Окрім цього, ключовими факторами є ефективність мережі, порядок виконання транзакцій на кожному вузлі, швидкість обробки операцій, а також реакція на запити, які надходять до смарт-контрактів.

Особливу увагу приділяють тестуванню масштабованості мережі та її здатності обробляти великий обсяг транзакцій. Це дозволяє виявити можливі проблеми у апаратному забезпеченні чи програмному забезпеченні перед впровадженням, а також дозволяє оцінити витрати на розгортання програм у хмарному середовищі. Тестування також орієнтоване на створення сценаріїв для оцінки загальної продуктивності блокчейн-системи.

Не менш важливою метою тестування є виявлення потенційних вразливостей додатків перед можливими атаками, а також перевірка надійності систем авторизації та аутентифікації. Тестування дозволяє виявити проблемні місця та забезпечити стабільність та надійність роботи блокчейн-системи в умовах реального використання.

Тестування безпеки у контексті блокчейн-систем включає ряд аспектів, таких як конфіденційність, цілісність, коректна робота сервера та доступність. Особлива увага приділяється виявленню потенційних порушень ідентифікаційного рівня у блокчейн-додатках. У разі виявлення таких проблем, транзакції, що вже знаходяться у виконанні, не можуть бути негайно зупинені. Це підкреслює важливість систематичного тестування безпеки з метою виявлення потенційних проблем у рівні ідентифікації.

Тестування безпеки у блокчейн-додатках також охоплює перевірку різних складових, таких як методи підпису крипто-гаманців, приватні ключі, алгоритми консенсусу та платформозалежні компоненти. Ще однією ключовою частиною є виявлення та запобігання шахрайським транзакціям, оскільки у технології блокчейн відміна транзакцій є майже неможливою.

Тестування на проникнення у блокчейн-системах дає можливість створювати та виконувати сценарії, що імітують кібератаки, для перевірки реакції системи на несанкціоновані спроби доступу. Цей підхід охоплює різні сценарії атак через мережеві сервіси, бездротові мережі, соціальну інженерію та блокчейн-додатки.

Забезпечення якості та тестування є критично важливими для впровадження блокчейн-систем. Безпосередні інциденти можуть мати серйозні наслідки, особливо у фінансовій сфері. Тому фокус тестування включає розгляд функціональних аспектів та областей, які потребують тестування у блокчейн-додатках, включаючи рекомендації стосовно фреймворків для автоматизації тестування, які спрямовані на ефективне проведення цих процедур.

3.3.3 Інструменти для тестування блокчейн веб-застосунків

Процес вибору інструментів для тестування блокчейн застосунку є важливим етапом у розробці, оскільки блокчейн додатки мають свої особливості, які потребують спеціалізованих підходів до тестування.

Рисунок 3.7 – Процес вибору інструментів для тестування

Згідно рисунку 3.7, першим кроком у виборі інструментів є саме вибір платформи блокчейну, на якій буде базуватися ваш застосунок. Ethereum, Hyperledger Fabric, Corda, або інша. Вибір платформи визначить можливості і інструменти тестування, які будуть доступні. В цій роботі було обрано Ethereum.

Наступним кроком необхідно ідентифікувати ключові функції блокчейн додатку, які потрібно тестувати. До цього списку належать смарт-контракти, механізми консенсусу, транзакції, інтеграції з гаманцями тощо.

Далі йде вибір самих інструментів. Основні інструменти тестування для блокчейн додатків в мережі Ethereum включають:

- Truffle Suite: Для тестування смарт-контрактів Ethereum.
- Ganache: Покращений емулятор Ethereum блокчейну для тестування.
- Remix: Веб-інтерфейс для створення, тестування та відлагодження смарт-контрактів.
- Geth: Клієнт Ethereum для тестування мережі.
- Web3.js, Ethers.js: Бібліотеки для взаємодії з Ethereum.

Після вибору інструментів, відбувається їхнє налаштування, а саме конфігурація бібліотек, опцій, видів тестування, підключення до пайплайну проєкту, щоб запускати тести автоматично і, відповідно, написання необхідних тестів для максимального покриття системи.

Процес тестування може включати юніт-тести смарт-контрактів, інтеграційні тести, тести з навантаженням та інші форми тестів для перевірки функціональності, безпеки та швидкодії вашого додатку.

3.3.4 Аналіз процесів виявлення та виправлення помилок

Виникнення помилок в системі веде за собою додаткові процеси виправлення та валідації компонентів системи. Поняття критичності та пріоритетності помилок надають раціональності в прийнятті рішення стосовно черговості виправлення помилок в системі та протягом процесу неперервної розробки(рис. 3.8).

Пріоритетність помилок вказує на важливість виправлення проблеми з точки зору бізнесу або користувача. Вона може бути визначена за такими критеріями, як вплив на функціональність програми, кількість користувачів, які її зазнають, чи величина втрати даних чи фінансів через цей баг.

Критичність багу визначає, наскільки серйозно він впливає на роботу програми. Це може бути визначено його впливом на систему в цілому, ймовірністю

виникнення проблеми, складністю виправлення та його впливом на безпеку програми.

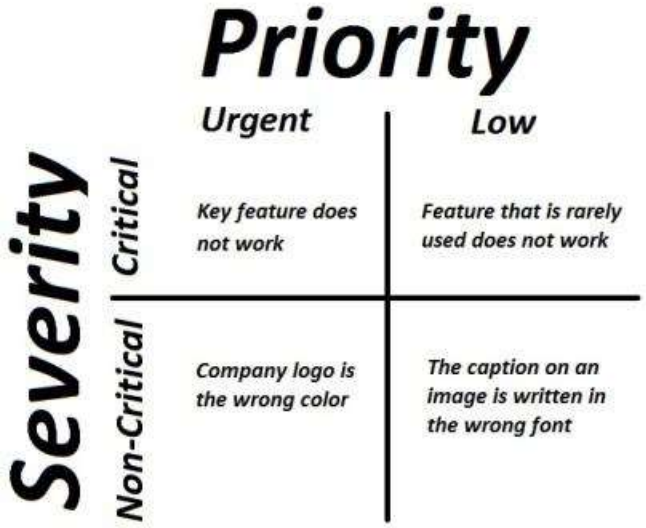


Рисунок 3.8 – Взаємозв'язок поняття критичності та пріоритетності багу

Нижче наведено приклади помилок та їх пріоритезацію.

Критичний баг: уразливість безпеки, яка дозволяє несанкціонованому користувачеві отримати доступ до приватної інформації. Пріоритет: вищий пріоритет через серйозний вплив на безпеку користувачів та системи.

Високий пріоритет: функціональна помилка, яка перешкоджає використанню важливої функції програми. Пріоритет: високий, оскільки ця помилка впливає на користувачів і перешкоджає нормальному використанню продукту.

Низький пріоритет: невелика граматична помилка в інтерфейсі користувача. Пріоритет: низький, оскільки ця помилка не впливає на роботу системи та не перешкоджає користувачам в основному використанні.

Пріоритезація помилок допомагає командам розробників ефективно керувати ресурсами та визначати послідовність виправлень для максимального задоволення потреб бізнесу та користувачів.

Окрім пріоритету виправлення багу, також варто оцінити вартість його виправлення. Фактори впливу на вартість багу можуть бути значною мірою визначені характером, складністю та впливом знайденого дефекту на функціонування

системи. На рисунку 3.9 зображений графік відношень вартості виправлення помилок в залежності від моменту їх виявлення.

Нижче наведено ключові фактори, що впливають на вартість багу:

- Етап тестування. Залежно від того, на якому етапі життєвого циклу проекту був виявлений баг, його вартість може змінюватися. Наприклад, якщо помилку було виявлено на етапі передрелізного тестування, виправлення його буде менш коштовним порівняно з виявленням на етапі випуску вже готової продукції.

Складність виправлення багу з технічної точки зору може значно впливати на його вартість. Чим складніше виявлений дефект, тим більше часу та зусиль потрібно для його виправлення, що може підвищити вартість

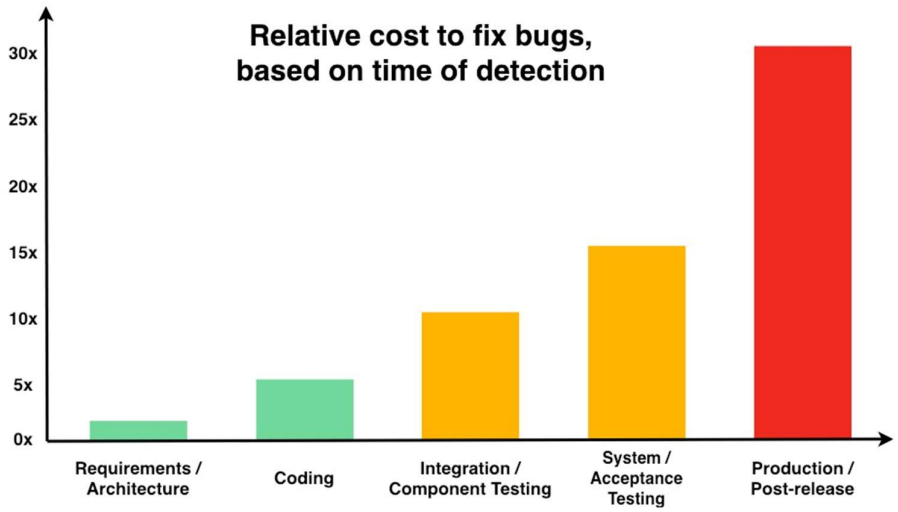


Рисунок 3.9 - Відносна вартість виправлення помилок залежно від часу виявлення

- Час, витрачений на фікс та перевірку. Вартість виправлення також залежить від часу, який розробники витрачають на виявлення, виправлення та перевірку багу. Кожна година, витрачена на виправлення, може вплинути на вартість роботи.

- Складність перевірки фіксу. Чим складніше перевіряти виправлення, тим більше часу та зусиль потрібно для забезпечення його коректності.

- Необхідність додаткових релізів. Якщо виправлення багу потребує додаткового випуску нової версії програмного забезпечення, це може збільшити вартість через додаткові зусилля з розгортки та підтримки нової версії.

Ці фактори можуть змінюватися в залежності від контексту конкретного проекту та характеристик самого багу. Оцінка вартості багу важлива для прийняття рішення щодо пріоритету та розподілу ресурсів у процесі розробки.

3.4 Автоматизація тестування блокчейн-застосунку.

Тестування блокчейн-застосунків вимагає великих зусиль та ресурсів через їх складну структуру та взаємодію з різними блокчейн-мережами.

Етапи автоматизації тестування блокчейн-застосунків:

1) Обрання підходу до автоматизації. Можна обрати між різними інструментами та методиками, такими як тестування вимог, функціональне тестування, тестування на відмови, тощо.

2) Підготовка тестової документації. Створення тестових сценаріїв, які охоплюють різні аспекти функціональності блокчейн-застосунків.

3) Вибір інструментів. В цій роботі обрані такі інструменти, як Cypress, Truffle, Ganache, Remix для тестування smart-контрактів.

4) Створення та запуск тестів.

5) Автоматизація та виконання. Реалізація автоматичного виконання тестів з використанням обраних інструментів та платформ для тестування.

Приклади завдань, які вирішує автоматизація тестування:

- Тестування Smart Contracts: Автоматизовані тести можуть перевірити працездатність та безпеку Smart Contracts.

- Функціональне тестування додатку: Автоматичні тести можуть перевірити функціональність різних частин додатку: реєстрацію користувача, транзакції, взаємодію з мережею тощо.

- Тестування відмов: Створення тестових сценаріїв, які перевіряють реакцію системи на відмови та непередбачувані умови.

Автоматизація тестування блокчейн-застосунків - це важливий крок для забезпечення якості та надійності програмного забезпечення, що використовується в галузі блокчейну. Забезпечуючи швидке виявлення помилок та забезпечення ефективного функціонування додатків, автоматизація тестування стає ключовим елементом в розробці блокчейн-застосунків.

В ході розробки даної системи було також створено та запущено на регулярній основі автоматизовані тести смарт-контрактів. На рисунку 3.10 Зображено успішне проходження створених юніт-тестів для перевірки роботи смарт-контрактів. В додатку Г наведено розроблені юніт-тести в даній роботі для створеного смарт-контракту.

```
C:\Users\tiwar\VSCodeWorkspace\Truffle-project-1>truffle test

Compiling your contracts...
===========================
√ Fetching solc version list from solc-bin. Attempt #1
√ Downloading compiler. Attempt #1.
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\SimpleStorage.sol
> Artifacts written to C:\Users\tiwar\AppData\Local\Temp\test--18112-6sMmaFPiOEUz
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang


  Contract: SimpleStorage
    √ Should update data (2189ms)


  1 passing (2s)
```

Рисунок 3.10 – Приклад успішного проходження юніт-тестів за допомогою інструменту Truffle

3.5 Аудит смарт-контрактів

Аудит смарт-контрактів - це процес оцінки безпеки, правильності реалізації функцій, відповідності до вимог та виявлення потенційних уразливостей в смарт-контрактах, які використовуються у блокчейні. Це критичний етап перед

впровадженням смарт-контрактів в продуктивне середовище, оскільки помилки в контрактах можуть призвести до втрати коштів, порушення безпеки та інших серйозних проблем. На рисунку 3.11 наведено етапи проведення аудиту смарт-контрактів.

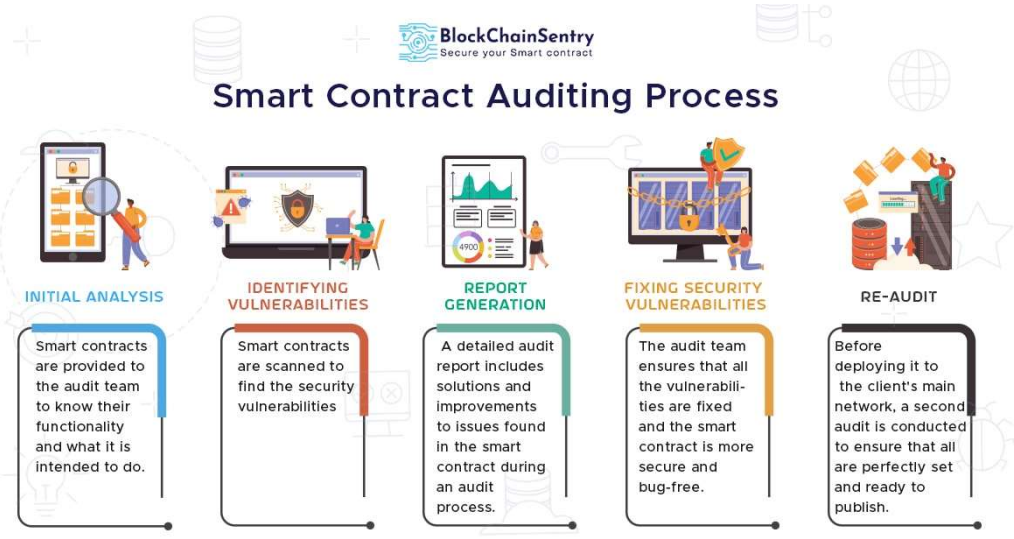


Рисунок 3.11 – Етапи проведення аудиту смарт-контрактів

Етапи аудиту смарт-контрактів включають:

1) Аналіз вимог: Ретельне вивчення функціональних вимог до контракту, його цілей та передбачуваних сценаріїв використання.

2) Технічний аналіз: Перевірка коду смарт-контракту на предмет відповідності кращим практикам, оптимізації, ефективності, відсутності потенційних уразливостей та недоліків.

3) Тестування безпеки: Запуск пенетраційного тестування, відмовостійкості та виявлення вразливостей, що можуть бути використані для атак або злому.

4) Підготовка звіту: Після аудиту складається детальний звіт, в якому фіксуються виявлені проблеми, поради щодо їх виправлення та рекомендації з покращення.

5) Виправлення знайдених вразливостей та повторний аудит.

Даний процес повторюється, поки власники програмного забезпечення не отримають задовільний результат у звіті аудиту [38].

Основні особи, відповідальні за аудит:

1) Аудитори: Спеціалісти з великим досвідом в розробці смарт-контрактів, криптографії та кібербезпеки, які виконують технічний аналіз та тестування безпеки контрактів.

2) Розробники: Команда, що створює смарт-контракти, надає необхідну документацію та надає відповіді на запитання аудиторів.

Цілі аудиту смарт-контрактів включають:

1) Визначення відповідності контракту вимогам та цілям.

2) Виявлення потенційних уразливостей та помилок у контракті.

3) Забезпечення безпеки, цілісності та стабільності контракту перед впровадженням в блокчейн.

4) Надання рекомендацій та порад щодо виправлення виявлених проблем.

В ході розробки даного застосунку було проведено аудит смарт-контракту, який лежить в основі застосунку і визначає логіку транзакцій в рамках нашого застосунку. На рисунку 3.12 зображено тести, які використовуються для аудиту за допомогою інструменту Echidna.

```
Echidna Test Run Result:

Property 1: Check that the buyer receives tokens after purchasing
  Test Case 1: PASS - Buyer's token balance increased after the purchase
  Test Case 2: PASS - Tokens transferred correctly to the buyer
  ...

Property 2: Ensure that the token price is always greater than zero
  Test Case 1: PASS - Token prices are always greater than zero
  Test Case 2: PASS - Price set for each token is greater than zero
  ...

Property 3: Verify that the owner receives a payment after a purchase
  Test Case 1: PASS - Contract owner received payment after purchase
  Test Case 2: PASS - Correct payment amount sent to the contract owner
  ...
```

Рисунок 3.12 – Результат виконання тестів аудиту за допомогою інструменту Echidna

3.6 Висновок до розділу

В процесі створення блокчейн-орієнтованих веб-застосунків виявлено необхідність ретельного розгляду певних аспектів. Одним з них є розробка інтерфейсу, де важливо створити інтуїтивно зрозумілі та функціональні елементи для зручного взаємодії користувачів з системою.

Також, вирішальне значення має належна організація процесу розробки та тестування. Оптимізація CI Pipeline дозволяє автоматизувати рутинні процеси та забезпечує безперервну поставку програмного забезпечення.

Процес забезпечення якості включає в себе планування тестування, розгляд різних видів тестування для забезпечення функціональності та безпеки системи, а також використання спеціалізованих інструментів для тестування блокчейн-веб-застосунків.

Усі ці аспекти, поєднані разом, допомагають забезпечити якісну розробку та стабільну роботу веб-застосунків на основі технології блокчейн. Систематична розробка і надійне тестування визначають успішність та функціональність цих застосунків в умовах постійних змін технологічного середовища.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Технологічний аудит розробленого веб застосунку

Як було зазначено раніше, популярність блокчейн-технологій пов’язана з їхньою надзвичайною безпекою даних, децентралізованістю, неможливістю змінювати або видаляти інформацію, швидкістю виконання транзакцій між користувачами блокчейну та іншими факторами. Як наслідок, мережа блокчейн ідеально підходить для виконання завдань, у яких основним завданням є забезпечення надійності та захисту даних.

Тому перед виконаною магістерською кваліфікаційною роботою було поставлено мету: використовуючи сучасні блокчейн-технології суттєво покращити процес надання послуг з продажу і покупок та підвищити рівень їх надійності.

Для цього нами було: детально вивчено мережі блокчейн та їх можливе використання; досліджено ринок на предмет виявлення різних способів надання послуг або товарів, де можна застосувати блокчейн-технології; спроектовано програмну частину додатку; розроблено модулі додатків та змодельовано базу даних; проведено тестування та розроблено тестову документацію для покриття всіх модулів додатку.

В результаті було розроблено додаток з відкритим кодом та якісно документованим програмним інтерфейсом (як окремим сервісом), що може бути використаний при створенні або інтеграції блокчейн-технології в уже існуючі онлайн платформи для підвищення надійності надання послуг продажу та покупок.

Для встановлення комерційного потенціалу розробленого нами мобільного додатка та програмного інтерфейсу було запрошено 3-х відомих експертів – кандидатів технічних наук, доцентів Кабачія В.В., Гармаша В.В. та Барабан М.В.

Встановлення комерційного потенціалу розробленого мобільного додатка та програмного інтерфейсу було здійснено за критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання технічного рівня та комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили розроблений нами мобільний додаток та програмний інтерфейс таким чином (див. таблицю 4.2):

Таблиця 4.2 – Результати технологічного аудиту розробленого мобільного додатка (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Кабачій В.В.	Гармаш В.В.	Барабан М.В.
	Бали, що їх виставили експерти:		
1	4	3	3
2	3	3	4
3	4	3	4
4	3	3	4
5	4	3	3
6	3	4	4
7	3	3	3
8	4	3	3
9	4	3	3
10	3	3	4
11	4	4	4
12	3	3	4
Сума балів	$СБ_1 = 42$	$СБ_2 = 38$	$СБ_3 = 43$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{42 + 38 + 43}{3} = \frac{123}{3} = 41,00$		

Встановлення комерційного потенціалу розробленого нами мобільного додатка та програмного інтерфейсу будемо здійснювати на основі рекомендацій, наведених в таблиці 4.3 [39].

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 41,0 балів, то це свідчить, що розроблений нами мобільний додаток та програмний інтерфейс має рівень комерційного потенціалу, який вважається «високим».

Це пояснюється тим, що використання технології блокчейн в системах реалізації механізму децентралізованого збереження та обробки даних в сфері послуг здійснюється за новим алгоритмом, в якому відбувається розподіл зберігання та шифрування даних, що допоможе уникнути ризиків шахрайства та потребуватиме набагато менше сторонніх сервісів для роботи.

4.2 Розрахунок витрат на розроблення мобільного додатка та програмного інтерфейсу

При виконанні роботи були зроблені певні витрати.

Зокрема:

А) Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \quad (4.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (6700…23000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 20 день;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	20000	1000	20 годин	≈ 3333
2. Магістрант-студент-виконавець	2000 (беремо 6700)	335	80	≈ 26800
3. Консультант з економічної частини	18000	900	1,5 години	≈ 225 (при 6-годинному робочому дні)
Загалом				$З_о$ = 30 358 грн

Б) Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о \qquad (4.2)$$

Приймемо, що α = 0,11. Тоді для випадку отримаємо:

$$З_д = 0{,}11 \times 30358 = 3339{,}38 \approx 3340 \text{ грн.}$$

В) Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (30358 + 3340) \times 0{,}22 = 7413{,}56 \approx 7414 \text{ грн.}$$

Г) Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для випадку можна прийняти, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 4.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	48000	25	3,1 (при 85% використанні	2635
2. Приміщення університету, кафедри	22000	3,5	3,1 при 90% використанні	≈ 179
Всього				А = 2814 грн

Д) Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.} \quad (4.5)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг; $Ц_i$ – вартість матеріалу *i*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$; $В_i$ – маса відходів матеріалу *i*-го найменування; $Ц_в$ – ціна відходів матеріалу *i*-го найменування; n – кількість видів матеріалів.

Е) Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн} \quad (4.6)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.; $Ц_i$ – ціна комплектуючих *i*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$; n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 1000 грн.

Ж) Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} \quad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,0 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 250 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0{,}9$.

$К_д$ – коефіцієнт корисної дії, $К_д = 0{,}8$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,0 \cdot 250 \cdot 0,9}{0,8} = 1265,62 \approx 1266 \text{ грн.}$$

И) Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о \quad (4.8)$$

Для випадку отримаємо:

$$В_{інш} = 1,2 \times 30358 = 36429,60 \approx 36430 \text{ грн.}$$

К) Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

$$В = 30358 + 3340 + 7414 + 2814 + 1000 + 1266 + 36430 = 82622 \text{ грн.}$$

Л) Загальні витрати на розроблення мобільного додатка та програмного інтерфейсу $В_{заг}$ становлять:

$$В_{заг} = \frac{В}{\beta} \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,90$, оскільки робота практично завершена.

Тоді:

$$В_{заг} = \frac{82622}{0,90} = 91802,22 \text{ грн або приблизно 92 тисячі грн.}$$

Тобто прогнозовані загальні витрати на розробку мобільного додатка та програмного інтерфейсу становлять приблизно 92 тисячі грн.

4.3 Розрахунок економічного ефекту від можливої комерціалізації розробки

Економічний ефект від впровадження та можливої комерціалізації розробленого нами мобільного додатка та програмного інтерфейсу пояснюється його значно кращими функціональними можливостями. Тому розробку можна реалізовувати на ринку дещо дорожче, ніж аналогічні за функціями розробки.

У 2022 році подібний за функціями мобільний додаток (але зі значно гіршими характеристиками) коштував приблизно 50 тисяч грн. Тоді такий «продвинутий» розроблений нами мобільний додаток та програмний інтерфейс можна буде реалізовувати на ринку в середньому приблизно за 70 тисяч грн або на 20 тисяч грн дорожче.

Аналіз ринку також показав, що потенційна кількість замовників такого мобільного додатка становить 6-8 осіб на рік. Для розрахунків приймемо 8 клієнтів. Разом з тим, проведений аналіз показав, що кількість таких клієнтів буде зростати, тобто можна очікувати зростання попиту на розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися протягом 2024-го, 2025-го та 2026-го років.

Прогноз зростання попиту на розробку складає по роках:

а) 2024 р. – приблизно +5 шт. до базового року;

б) 2025 р. – +15 шт. до базового року;

в) 2026 р. – +10 шт. до базового року (оскільки можуть з’явитися ще кращі розробки, зроблені студентами-магістрантами ВНТУ).

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від виведення розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}) \qquad (4.10)$$

де $\Delta Ц_о$ – покращення основного якісного показника від впровадження результатів розробки у цьому році. Для випадку це є збільшення ціни реалізації розробки $\Delta Ц_о = 70 - 50 = + 20$ тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 8 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки.

Таке покращення становитиме по роках, відповідно: у 2024 році – + 5 шт., у 2025 році + 15 шт., та у 2026 році + 10 шт.;

$Ц_о$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_о$= 70 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість;

$\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = (0{,}2...0{,}5)$; візьмемо $\rho = 0{,}5$;

υ – ставка податку на прибуток. У 2023-25 роках υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження розробки (2024 р.) становитиме:

$$\Delta П_1 = [20 \cdot 8 + 70 \cdot 5] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 174 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження розробки протягом другого (2025) року складе:

$$\Delta П_2 = [20 \cdot 8 + 70 \cdot 15] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 413 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від

можливого впровадження розробки протягом третього (2026) року складе:

$$\Delta П_3 = [20 \cdot 8 + 70 \cdot 10] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 294 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення мобільного додатка до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації розробки, складе:

$$ПП = \frac{174}{(1+0,1)^2} + \frac{413}{(1+0,1)^3} + \frac{294}{(1+0,1)^4} \approx 144 + 310 + 201 = 655 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV (вартість стартапу), що повинні бути вкладені для реалізації розробки: PV = (1,0…5) × $В_{заг}$.

Для випадку PV = (1,0…5) × 92 = 2 × 92 = 184 тисяч грн.

Абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV \quad (4.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливого впровадження розробки, грн;

PV – теперішня вартість інвестицій PV = 184 тисяч грн.

Абсолютний ефект від можливого впровадження розробки складе:

$$E_{абс} = 655 – 184 = 471 \text{ тисяч грн.}$$

Оскільки $E_{абс} > 0$, то комерціалізація розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $E_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.13)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 471 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 184 тис. грн;

$T_ж$ – життєвий цикл розробки, роки.

$T_ж$ = 4 років (2023-й, 2024-й, 2025-й, 2026-й роки)

Для випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{471}{184}} - 1 = \sqrt[4]{1 + 2,5598} - 1 = \sqrt[4]{3,5598} - 1 = 1,373 - 1 = 0,373 = 37,3\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f \qquad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022-2023 роках в Україні d = (0,10...0,12);

f – показник, що характеризує ризикованість вкладень;

f = (0,1...0,50). Приймемо f = 0,25.

Для випадку отримаємо:

$$\tau_{мін} = 0{,}12 + 0{,}25 = 0{,}37 \text{ або } \tau_{мін} = 37\%.$$

Оскільки величина $E_в = 37{,}3\% > \tau_{мін} = 37\%$, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробки.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленого нами мобільного та програмного інтерфейсу.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \qquad (4.15)$$

Для випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0{,}373} = 2{,}68 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленого нами мобільного додатка та програмного інтерфейсу.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні.

Якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{174}{(1+0{,}2)^2} + \frac{413}{(1+0{,}2)^3} + \frac{294}{(1+0{,}2)^4} \approx 121 + 239 + 142 = 502 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження розробки за три роки складе:

$$E_{абс} = 502 - 184 = 318 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.16)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 318 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 184 тисяч грн.

Для випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{318}{184}} - 1 = \sqrt[4]{1 + 1,7283} - 1 = \sqrt[4]{2,7283} - 1 = 1,285 - 1 = 0,285 = 28,5\%.$$

Оскільки величина $E_в$ = 28,5% < $\tau_{мін}$ = 37%, то потенційний інвестор може бути НЕ зацікавлений у фінансуванні та комерціалізації розробки.

Оскільки представлена до захисту магістерська кваліфікаційна роботи виконувалася двома студентами, і один з них – Русавський О.О. (гр. 1АКІТ-22м) зробив моделювання залежності величини внутрішньої дохідності потенційних $E_в$ інвестицій, вкладених у можливу комерціалізацію нашої розробки, від рівня інфляції в країні, а саме: при рівні інфляції в країні τ=10% внутрішня дохідність інвестицій становитиме 37,3%, а при рівні інфляції в країні τ = 20% внутрішня дохідність інвестицій становитиме 28,5%, то для отримання більш точних результатів додатково проведемо моделювання залежності величини внутрішньої дохідності потенційних $E_в$ інвестицій, вкладених у можливу комерціалізацію нашої розробки, від рівня інфляції в країні при τ = 5% та τ=30%.

Якщо рівень інфляції в країні буде 5% (що прогнозується на 2023 р.), то:

$$ПП = \frac{174}{(1+0,05)^2} + \frac{413}{(1+0,05)^3} + \frac{294}{(1+0,05)^4} \approx 158 + 357 + 242 = 757 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$Е_{абс} = 757 – 184 = 573 \text{ тисяч грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1,$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс} = 573$ тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 184 тисяч грн.

Для нашого випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{573}{184}} - 1 = \sqrt[4]{1 + 3,1141} - 1 = \sqrt[4]{4,1141} - 1 = 1,424 - 1 = 0,424 = 42,4\%.$$

Оскільки величина $Е_в = 42,4\% > \tau_{мін} = 37\%$, то потенційний інвестор може бути зацікавлений у фінансуванні та комерціалізації нашої розробки.

Якщо рівень інфляції в країні зросте до 30%, то:

$$ПП = \frac{174}{(1+0,3)^2} + \frac{413}{(1+0,3)^3} + \frac{294}{(1+0,3)^4} \approx 103 + 188 + 103 = 394 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$Е_{абс} = 394 – 184 = 210 \text{ тисяч грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 210 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 184 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{210}{184}} - 1 = \sqrt[4]{1+1,1413} - 1 = \sqrt[4]{2,1413} - 1 = 1,21 - 1 = 0,21 = 21,0\%.$$

Оскільки величина $E_в = 21,0\% < \tau_{мін} = 37\%$, то потенційний інвестор може мати сумніви у доцільності фінансування та комерціалізації нашої розробки.

Зроблені розрахунки у вигляді графіків наведено на рис. 5.1.

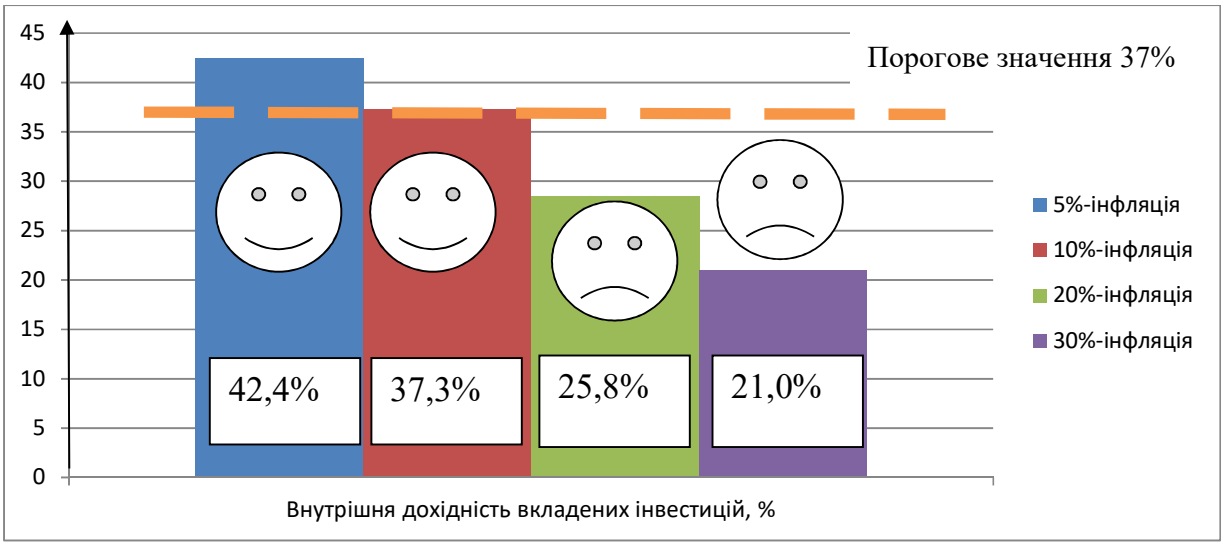


Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні

Аналіз діаграм на рис 5.1 показує, що при рівні інфляції в 5% та 10% величина внутрішньої дохідності інвестицій становить, відповідно, 42,4% та $E_в$ = 37,3%, що більше за порогове значення $\tau_{мін} = 37\%$ і тому комерціалізація нашої розробки може бути доцільною.

При рівні інфляції в 20% та 30% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію нашої розробки, становить, відповідно 25,8%

та 21%, що менше порогового значення $\tau_{мін} = 37\%$, і тому комерціалізація нашої розробки потенційним інвестором може бути проблематичною.

Остаточне рішення з цього питання потребує проведення додаткових розрахунків (можливо – зниження рівня ризикованості вкладень тощо).

ВИСНОВКИ

Всі задачі, поставлені перед магістерською дипломною роботою виконано в повному обсязі.

В даній магістерській кваліфікаційній роботі було розглянуто сукупність процесів та інструментів розробки веб-застосунку, що використовує технологію блокчейн. До сукупності процесів належать визначення вимог, дизайн, розробка, тестування та, власне, фіналізація продукту, а саме розміщення його на певних хмарних сховищах чи на приватних серверах. Досліджено основні компоненти блокчейн застосунків, які варто мати на увазі при розробці подібних додатків. В залежноті від мети та вимог до розробки, ці інструменти і компоненти дещо можуть змінюватися, але головні принципи описані в цій роботі зберігаються.

Розроблено графічний інтерфейс для наочної демонстрації прикладу інфраструктури блокчейн-проєкту. Головними розробленими компонентами є графічний інтерфейс самого сервісу, а також плагін для сторінок брендів, які в свою чергу, адаптують його під свої цілі.

Не менш важливою частиною даної роботи є саме організація пайплайну неперервної розробки, де основна увага була приділена процесам тестування користувацького інтерфейсу, а також тестування смарт-контрактів та транзакцій в мережі Ethereum. Окрмі цього, надано змістовний опис всіх етапів розробки життєвого циклу програмного забезпечення.

В якості новизни роботи слід відзначити застосування блокчейн-технологій у системах для децентралізованого зберігання та обробки інформації в сфері послуг, а також представлення нових етапів тестування веб-застосунків, які використовують блокчейн та особливості організації процесу тестування вцілому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. П. Кравченко. Вступ / П. Кравченко, Б. Скрябін, О. Дубініна / Блокчейн і децентралізовані системи – Харків, 2019 – 9 с.

2. Decentralized document version control using ethereum blockchain and IPFS [Електронний ресурс]: www.sciencedirect.com – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0045790618333093.

3. ДЕЦЕНТРАЛІЗОВАНІ ДОДАТКИ [Електронний ресурс]. – Режим доступу : https://avada-media.ua/ua/decentralizovannie-prilojeniya/.

4. ПЕРСПЕКТИВИ РОЗВИТКУ ДОДАТКІВ БЛОКЧЕЙН В УКРАЇНІ [Електронний ресурс]. – Режим доступу : https://csecurity.kubg.edu.ua/index.php/journal/article/view/18.

5. Global perspectives on blockchain adoption by industry [Електронний ресурс]. – Режим доступу : https://www2.deloitte.com/jp/en/pages/financial-services/articles/bk/blockchain-adoption-by-industry.html.

6. A Review on BlockChain Security [Електронний ресурс]. – Режим доступу : https://iopscience.iop.org/article/10.1088/1757-899X/396/1/012030/meta.

7. The Rise in Popularity of Cryptocurrency and Associated Criminal Activity [Електронний ресурс]. – Режим доступу : https://journals.sagepub.com/doi/full/10.1177/1057567719827051.

8. Що таке технологія блокчейн? [Електронний ресурс]. – Режим доступу : https://aws.amazon.com/ru/what-is/blockchain/.

9. Blockchain technology, bitcoin, and Ethereum [Електронний ресурс]. – Режим доступу : https://ieeexplore.ieee.org/abstract/document/8345547.

10. The Importance of Open Source for Blockchain Technology [Електронний ресурс]. – Режим доступу: https://www.koombea.com/blog/importance-open-source-blockchain-technology/.

11. Молодь в науці - дослідження, проблеми, перспективи [Електронний ресурс]. – Режим доступу https://conferences.vntu.edu.ua/index.php/mn/mn2023.

12. Melanie Swan. Інформаційні технології та їх технічна реалізація/ Melanie Swan // O'Reilly Media, 2015.

13. Daniel Drescher Blockchain Basics: A Non-Technical Introduction in 25 Steps/ Daniel Drescher // John Wiley & Sons, Inc. 2017.

14. Antony Lewis. The Basics of Bitcoins and Blockchains / Antony Lewis / Taylor & Francis Group. 2018

15. Overview and Opportunities of Blockchain Technology [Електронний ресурс]. – Режим доступу: https://www.cst.gov.sa/ar/Digitalknowledge/Documents/BlockchainDetailedStudyen.pdf

16. Rajkumar Buyya. Internet of Things: Principles and Paradigms/ Rajkumar Buyya, Amir Vahid Dastjerdi. – Elsevier Science, 2016. – 856 p. – ISBN 9780128093474

17. Minimum Viable Product (MVP) [Електронний ресурс]. – Режим доступу: https://www.productplan.com/glossary/minimum-viable-product/

18. What Is SDLC (Software Development Lifecycle) [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/sdlc/?nc1=h_ls

19. What is Blockchain? [Електронний ресурс]. – Режим доступу: https://lisk.io/what-is-blockchain.

20. A Review on BlockChain Security [Електронний ресурс]. – Режим доступу: https://iopscience.iop.org/article/10.1088/1757-899X/396/1/012030/meta.

21. Block hashing algorithm [Електронний ресурс]. – Режим доступу: https://en.bitcoin.it/wiki/Block_hashing_algorithm.

22. On Public and Private Blockchains. [Електронний ресурс]. – Режим доступу: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains

23. P. Deepak. Everything You Wanted to Know About the Blockchain/ P. Deepak, M. Nisha, and S. P. Mohanty. IEEE Consum. Electron. Mag., vol. 7, no. 4, pp. 6–14, 2018.

24. L. Luu, D. H. Chu. Making smart contracts smarter/, L. Luu, D. H. Chu, H. Olickel, P. Saxena, A. Hobor. Proceedings of the ACM Conference on Computer and

Communications Security, 2016, vol. 24-28-October-2016, pp. 254–269.

25. G. A. Oliva, A. E. Hassan, and Z. M. Jiang. An exploratory study of smart contracts in the Ethereum blockchain platform. - Empir. Softw. Eng., 2020, pp. 1–41.

26. The Ethereum Virtual Machine — How does it work? [Електронний ресурс]. – Режим доступу: https://medium.com/mycrypto/the-ethereum-virtual-machinehow-

does-it-work-9abac2b7c9e.

27. What Is Ethereum [Електронний ресурс] – Режим доступу: https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/ch01.html.

28. The benefits of PostgreSQL [Електронний ресурс] – Режим доступу: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql

29. What is TypeScript [Електронний ресурс] – Режим доступу: https://itjet.io/blog/what-is-typescript

30. Using Vue [Електронний ресурс] – Режим доступу: https://thecodest.co/blog/pros-and-cons-of-vue/

31. Що таке TDD і BDD і що повинен знати про них фронтендер [Електронний ресурс] – Режим доступу: https://senior.ua/articles/scho-take-tdd--bdd--scho-povinen-znati-pro-nih-frontender

32. User Interface Design Basics [Електронний ресурс] – Режим доступу: https://www.usability.gov/what-and-why/user-interface-design.html

33. Spillner A. Software Testing Foundations / Spillner A., Linz T., Schaefer H. – [4th Edition] – Rocky Nook, 2014. – 305 p. – ISBN 978-1-937538-42-2

34. What is a CI/CD pipeline? [Електронний ресурс] – Режим доступу: https://www.redhat.com/en/topics/devops/what-cicd-pipeline

35. Lina Wang. Application of Blockchain in Life Cycle Cost Management of Weapon Equipment / Lina Wang, Yanhong Zhang, Ying Ren. - 2020 2nd International Conference on Applied Machine Learning (ICAML). – 2020;

36. Barbara Thompson. Blockchain Testing Tutorial [Електронний ресурс] Режим доступу до ресурсу: https://www.guru99.com/blockchain-testing.html

37. Software Testing News. The need for Blockchain Testing [Електронний ресурс] Режим доступу до ресурсу: https://www.softwaretestingnews.co.uk/the-need-for-blockchain-testing/

38. How To Audit A Smart Contract: A Deep Dive Into Hacken's Process [Електронний ресурс] Режим доступу до ресурсу: https://hacken.io/discover/smart-contract-audit-process/

39. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

Додатки

94

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ
д.т.н., проф. Бісікало О. В
«12» 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Розробка архітектури, інтерфейсу та забезпечення якості продукту для ін-те-грації блокчейн технології у сфері послуг»

08-31.МКР.006.02.000 ТЗ

Керівник роботи
д.т.н., проф.
Роман КВЄТНИЙ
«12» 10 2023 р.

Виконавець:
ст. гр. 1ІСТ-22М
Володимир ЖИГАНОВ
«12» 10 2023 р.

Вінниця ВНТУ 2023

1. Назва та галузь застосування

Магістерська кваліфікаційна робота: «Розробка архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг». Галузь застосування – інформаційні технології.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету№247 від 18.09.2023 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є розробка інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг.

4. Джерела розробки

1. Global perspectives on blockchain adoption by industry [Електронний ресурс]. – Режим доступу : https://www2.deloitte.com/jp/en/pages/financial-services/articles/bk/blockchain-adoption-by-industry.html.

2. P. Deepak. Everything You Wanted to Know About the Blockchain/ P. Deepak, M. Nisha, and S. P. Mohanty. IEEE Consum. Electron. Mag., vol. 7, no. 4, pp. 6–14, 2018.

3. Spillner A. Software Testing Foundations / Spillner A., Linz T., Schaefer H. – [4th Edition] – Rocky Nook, 2014. – 305 p. – ISBN 978-1-937538-42-2

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система Windows 10 або Ubuntu 16 і вище; доступ до інтернету; оперативна пам’ять 4Гб; процесор intel core i5.

Результати роботи програми: купівля НФТ токену; валідація токену на

сайті бренду для надання дозволу купівлі преміум товару; виведення списку нфт токенів, які має користувач; виведення списку колекцій від різних брендів; виведення списку брендів.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – не більше 100 тис. грн;
- абсолютний ефект від впровадження розробки – не менше 450 тис. грн;
- внутрішня дохідність інвестицій – не менше 37%;
- термін окупності – не більше 3 років.

7. Стадії розробки

а) Аналіз предметної області	20.09-03.10
б) Вибір оптимальних інформаційних технологій	03.10-06.10
в) Вибір мови програмування та середовища розробки	06.10-12.10
г) Проектування та програмна реалізація	12.10-10.11
д) Економічне обґрунтування	11.11-16.11
е) Оформлення матеріалів до захисту МКР	17.11-20.11

8. Порядок контролю та приймання

Рубіжний контроль провести до 07.12.2023.

Попередній захист магістерської кваліфікаційної роботи провести до 21.11.2023.

Захист магістерської кваліфікаційної роботи провести в період з 18.12.2023 до 19.12.2023.

Додаток Б (обов'язковий)

Ілюстративна частина

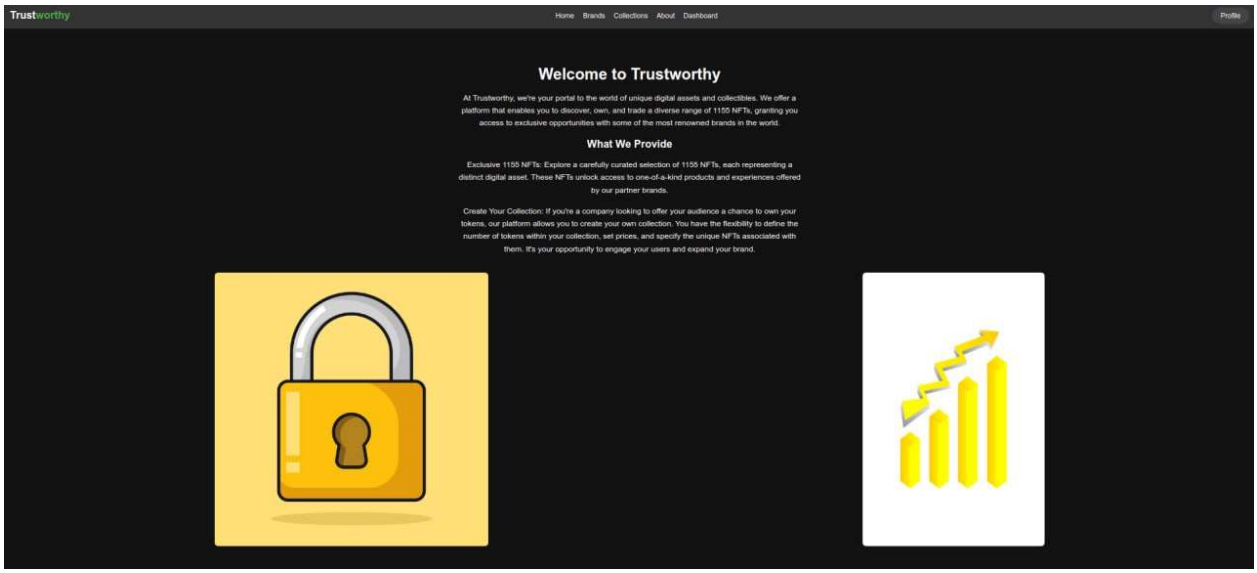


Рисунок Б.1 – Вигляд сторінки Home

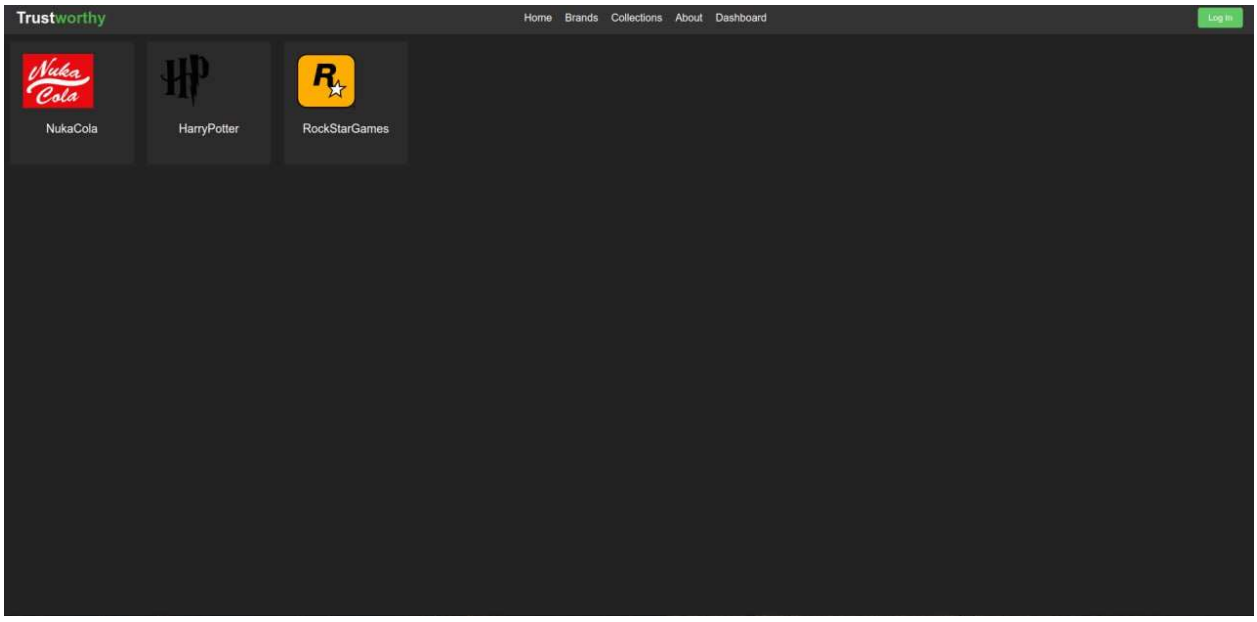


Рисунок Б.2 – Вигляд сторінки Brands

Продовження додатку Б

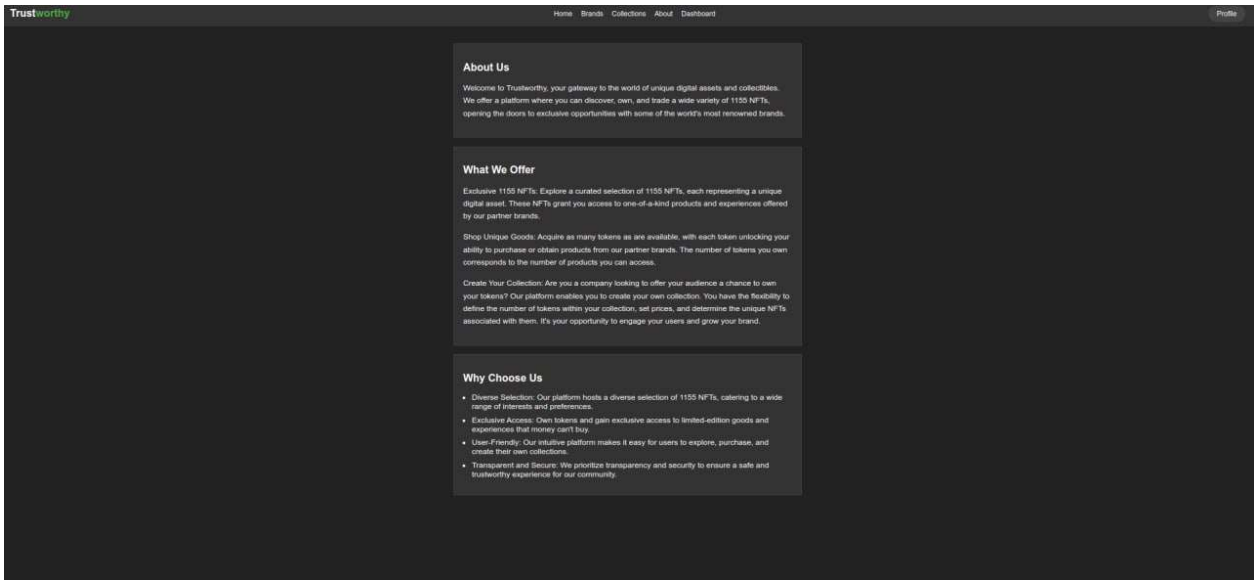


Рисунок Б.3 – Вигляд сторінки About

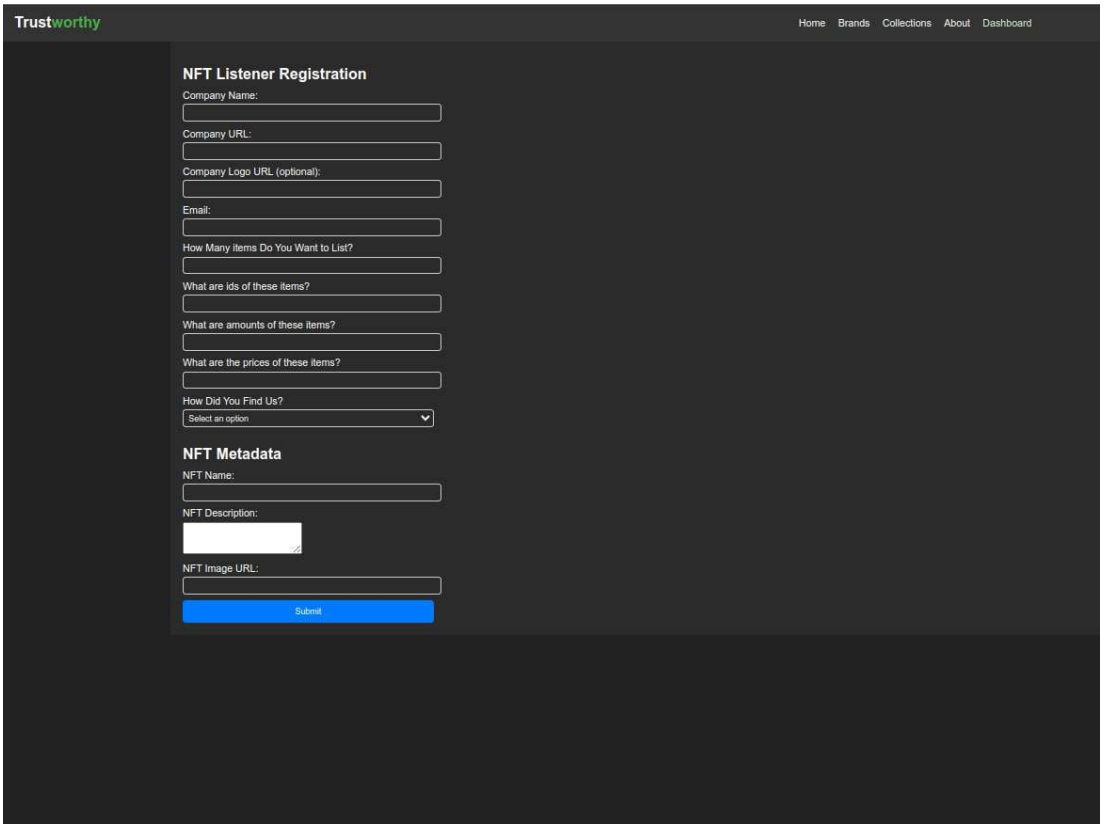


Рисунок Б.4 – Вигляд сторінки Dashboard

Продовження додатку Б

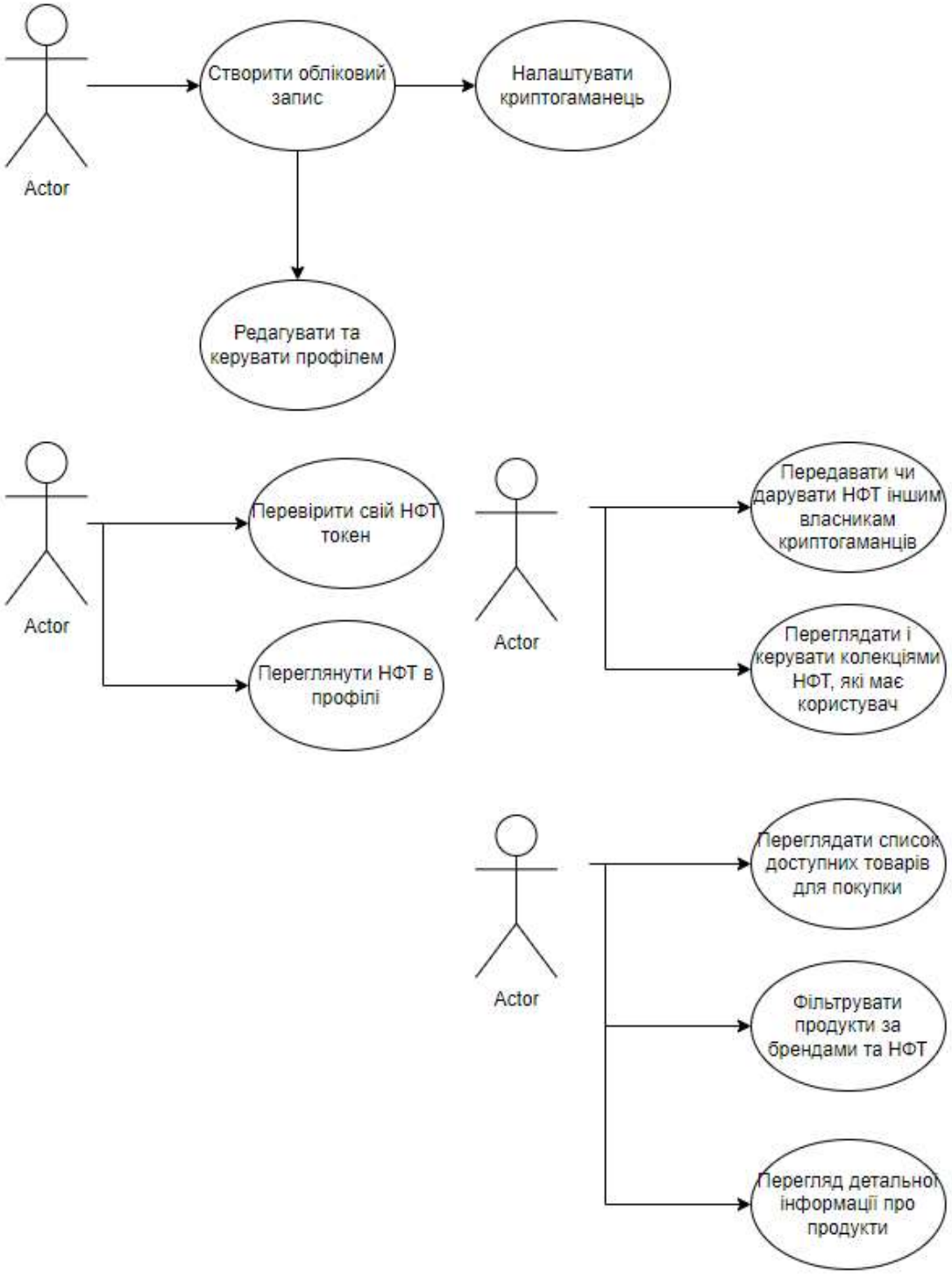


Рисунок Б.5 – Схема взаємодій користувача згідно вимог

Продовження додатку Б

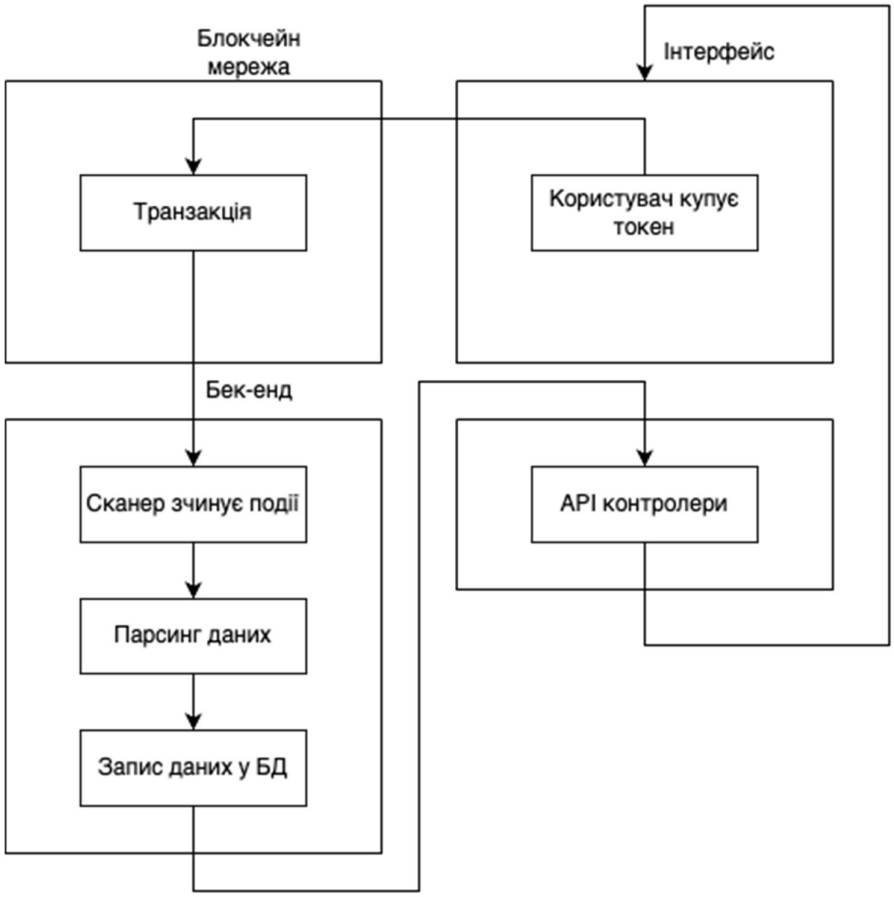


Рисунок Б.6 - Схема загальної роботи застосунку

Рисунок Б.7 - Схема загальної роботи плагіну

Додаток В (обов'язковий)

Лістинг розроблених тестів

Покриття функціоналу смарт-контракту задопомогою Truffle:

```
// File: test/TWCustomerCollectionTest.sol
pragma solidity ^0.8.0;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/TWCustomerCollection.sol";

contract TWCustomerCollectionTest {
    TWCustomerCollection twCustomerCollection;
    address owner;

    // Set up the contract instance before each test
    function beforeEach() public {
        owner = address(this); // Using 'this' address as owner for
testing purposes
        uint256[] memory tokenIds = new uint256[](3);
        uint256[] memory initialAmounts = new uint256[](3);
        tokenIds[0] = 1;
        tokenIds[1] = 2;
        tokenIds[2] = 3;
        initialAmounts[0] = 100;
        initialAmounts[1] = 200;
        initialAmounts[2] = 300;

        twCustomerCollection   =   new   TWCustomerCollection("Test
Collection", "https://example.com/api/", tokenIds, initialAmounts);
    }

    // Test case to check contract initialization values
    function testContractInitialization() public {
        Assert.equal(twCustomerCollection.name(),            "Test
Collection", "Contract name should match");
        Assert.equal(twCustomerCollection.uri(),
"https://example.com/api/", "Contract URI should match");

        uint256[] memory expectedTokenIds = new uint256[](3);
        expectedTokenIds[0] = 1;
        expectedTokenIds[1] = 2;
        expectedTokenIds[2] = 3;

        uint256[] memory expectedInitialAmounts = new uint256[](3);
        expectedInitialAmounts[0] = 100;
        expectedInitialAmounts[1] = 200;
        expectedInitialAmounts[2] = 300;

        Assert.equal(twCustomerCollection.tokenIds(0),
```

```
expectedTokenIds[0], "Token ID 1 should match");
        Assert.equal(twCustomerCollection.initialAmounts(0),
expectedInitialAmounts[0], "Initial amount for Token ID 1 should
match");
    }

    // Test case to check minting of initial tokens by owner
    function testMintInitTokens() public {
        twCustomerCollection.mintInitTokens();

        Assert.equal(twCustomerCollection.balanceOf(owner, 1), 100,
"Balance of Token ID 1 should match");
        Assert.equal(twCustomerCollection.balanceOf(owner, 2), 200,
"Balance of Token ID 2 should match");
        Assert.equal(twCustomerCollection.balanceOf(owner, 3), 300,
"Balance of Token ID 3 should match");
    }
}
```

Автоматизовані тести для аудиту смарт-контракту:

```
pragma solidity 0.4.24;

import "contracts/TWCustomerCollection.sol";

contract TEST is TWCustomerCollection {
    uint256 initial_totalSupply;
    address public deployer;
    address               public               user1               =
0x00a329c0648769a73afac7f9381e08fb43dbea40;
    address               public               user2               =
0x00a329c0648769a73afac7f9381e08fb43dbea50;
    address               public              policy               =
0x00a329c0648769a73afac7f9381e08fb43dbea60;
    uint256 constant public INITIAL_SUPPLY = 50000000000000000;

    constructor() public {
        deployer = msg.sender;
        initialize(deployer);
        _monetaryPolicy = policy;
        transfer(user1, 750000000000);
        transfer(user2, 250000000000);
        initial_totalSupply = _totalSupply;
    }

    function other_user() internal returns (address) {
        if (msg.sender == user2) {
            return user1;
        } else {
            return user2;
        }
```

```
    }

    function echidna_balanceToZero() public returns (bool) {
        if(_gonBalances[msg.sender] == 0) {
            return true;
        }
        return balanceOf(msg.sender) != 0;
    }

    function echidna_no_burn() returns (bool) {
        return (_gonBalances[0x0] == 0);
    }

    function echidna_self_transfer() returns (bool) {
        uint balance = _gonBalances[msg.sender].div(_gonsPerFrag-
ment);
        return (transfer(msg.sender,balance));
    }

    function echidna_zero_transfer() returns (bool) {
        return (transfer(other_user(),0));
    }

    function echidna_gons_invariant() returns (bool) {
        return (_gonsPerFragment == TOTAL_GONS.div(_totalSupply));
    }

    function echidna_self_approve_and_self_transferFrom() returns
(bool) {
        uint balance = _gonBalances[msg.sender].div(_gonsPerFrag-
ment);
        approve(msg.sender, 0);
        approve(msg.sender, balance);
        return (transferFrom(msg.sender,msg.sender,balance));
    }

    function echidna_self_approve_and_transferFrom() returns (bool)
{
        uint balance = _gonBalances[msg.sender].div(_gonsPerFrag-
ment);
        approve(msg.sender, 0);
        approve(msg.sender, balance);
        return (transferFrom(msg.sender,other_user(),balance));
    }
}
```

Автоматичні тести для інтерфейсу користувача:

```
describe('Blockchain Application Tests', () => {
    beforeEach(() => {
        // Assuming the application starts from a certain page or
login state
        cy.visit(url);
```

```
    });

    it('should display available tokens on the platform', () => {
        // Test to ensure that available tokens are displayed
correctly
        cy.get('.token-list').should('have.length.greaterThan', 0);
// Replace '.token-list' with your token list selector
    });

    it('should allow a user to purchase tokens', () => {
        // Test the process of buying tokens
        // Click on a token to initiate the purchase process
        cy.get('.token-item').first().click(); // Replace '.token-
item' with your token item selector

        // Verify that the purchase modal or form is displayed
        cy.get('.purchase-modal').should('be.visible'); // Replace
'.purchase-modal' with your purchase modal selector

        // Fill in the purchase form fields
        cy.get('input[name=amount]').type('5');  //  Assuming  the
input field for token amount
        cy.get('button[data-testid=purchase-button]').click();   //
Replace 'data-testid' with your purchase button test ID

        // Check for a success message or confirmation of the
purchase
        cy.contains('.success-message',                    'Purchase
successful').should('be.visible');  //  Replace  '.success-message'
with your success message selector
    });

    it('should update user balance after a successful purchase', ()
=> {
        let initialBalance;
        cy.get('.user-balance').invoke('text').then((text) => {
            initialBalance = parseFloat(text); // Assuming the user
balance is displayed in '.user-balance'
        });
        // Perform a token purchase
        cy.get('.token-item').first().click();
        cy.get('input[name=amount]').type('5');
        cy.get('button[data-testid=purchase-button]').click();

        // Check if the user's balance is updated after the purchase
        cy.get('.user-balance').invoke('text').then((text) => {
            const updatedBalance = parseFloat(text);
            expect(updatedBalance).to.eq(initialBalance - 5);
        });
    });
    it('should allow a user to purchase NFT tokens', () => {
            // Test the process of buying NFT tokens
            // Click on an NFT item to initiate the purchase process
```

```
            cy.get('.nft-item').first().click(); // Replace '.nft-
item' with your NFT item selector
            // Verify the purchase modal or form for NFT is displayed
            cy.get('.purchase-nft-modal').should('be.visible'); //
Replace '.purchase-nft-modal' with your NFT purchase modal selector

            // Fill in the purchase form fields for NFT
            cy.get('input[name=nft-id]').type('123'); // Assuming
the input field for NFT ID
            cy.get('button[data-testid=purchase-nft-
button]').click(); // Replace 'data-testid' with your NFT purchase
button test ID
        // Check for a success message or confirmation of the NFT
purchase
        cy.contains('.success-message', 'NFT purchase
successful').should('be.visible'); // Replace '.success-message'
with your success message selector
        });

    it('should allow navigation to the About Us page', () => {
        // Test navigation to the About Us page
        cy.contains('About Us').click(); // Assuming the link text
for the About Us page

        // Validate if the About Us page is loaded
        cy.url().should('include', '/about-us'); // Replace
'/about-us' with the actual URL of the About Us page
        cy.get('h1').should('contain', 'About Us'); // Replace 'h1'
with the heading element of the About Us page
        });

    it('should validate page contents and elements', () => {
        // Validate presence of key elements on the home page
        cy.get('.header').should('exist'); // Replace '.header'
with your header element selector
        cy.get('.token-list').should('exist'); // Replace '.token-
list' with your token list selector
        cy.get('.footer').should('exist');
});
```

Додаток Г

Довідка про впровадження результатів роботи

FIVE
BUILDING THE FUTURE

21027, Україна, м.Вінниця, просп.Космонавтів 42,кв.3,
тел. +380673490106, www.fivesysdev.com

За місцем вимоги

Довідка

Видана _____ в тому, що результати, одержані ним в процесі виконання магістерської дипломної роботи «Розробка архітектури, інтерфейсу та забезпечення якості продукту для інтеграції блокчейн технології у сфері послуг», а саме графічний інтерфейс, тестова документація та автоматизовані тести, планується використати в розробках ТОВ «ФАЙВ Системз Девелопмент».

Директор _____ Терещенко С.М.
(підпис) (прізвище та ініціали)

Товариство з обмеженою відповідальністю «ФАЙВ СИСТЕМЗ ДЕВЕЛОПМЕНТ» 39859344 Україна м.Вінниця

107

Додаток Д (обов'язковий)

Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка архітектури, інтерфейсу та забезпечення якості продукту для інте-грації блокчейн технології у сфері послуг

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації
(кафедра, факультет)

Показники звіту подібності Unicheck

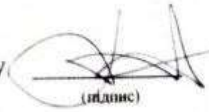
Оригінальність 99.3% Схожість 0.7%

Аналіз звіту подібності (відмітити потрібне):

☑ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

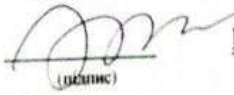
☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку  Роман МАСЛІЙ
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи  Володимир ЖИГАНОВ
(підпис)

Керівник роботи  Роман КВЄТНИЙ
(підпис)