Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Розробка NFT маркетплейсу
Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
Старший викладач
Гороховський К.С.
_____ (Підпис)
"___" _____ 2024 року
Виконала студентка
КН-3 Луцюк І. Р.
"___" _____ 2024 року

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

Жежерун Олександр Петрович

„_____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Луцюк Іванна 3-го курсу, факультету інформатики
ТЕМА : Розробка NFT маркетплейсу

Зміст ТЧ до курсової роботи :

Зміст
Анотація
Вступ
Розділ 1. Технічні відомості
Розділ 2. Невзаємозамінні токени або NFT
Розділ 3. Деталі розробки системи
Висновки
Додатки

Дата видачі „___” _____ 2021 р.

Керівник_____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи

№ з/п	Назва Етапу	Термін Виконання	Примітка
1	Вибір теми	05.10.2023	
2	Затвердження теми з керівником	20.10.2023	
3	Дослідження обраної технології. Вивчення інструментів розробки	01.02.2024	
4	Розробка застосунку	20.03.2024	
5	Написання текстової частини	13.04.2024	
6	Захист курсової роботи	20.05.2024	

Студентка Луцюк І.Р

Керівник Гороховський К.С. “_____”_____ 2024

Зміст

Перелік прийнятих скорочень

HTML - HyperText Markup Language

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

HTTP - HyperText Transfer Protocol

DLT - Distributed Ledger Technology

NFT - Non-Fungible Token

BTC - Bitcoin

PoW - Proof-of-Work

PoS - Proof-of-Stake

PoA - Proof-of-Authority

JSON - JavaScript Object Notation

ERC - Ethereum Request for Comments

EVM - Ethereum Virtual Machine

DApp - Decentralized Application

IPFS - InterPlanetary File System

CSV - Comma-Separated Values

CID - Content Identifier

SDK - Software Development Kit

Анотація

У даній роботі було розроблено маркетплейс платформу для безпечної купівлі та продажу невзаємозамінних токенів на базі блокчейн. Маркетплейс надає можливість безпечно купувати та продавати NFT, а також брати участь в аукціонах, завдяки авторизації через Metamask гаманець. Розробка цієї системи була за використання Polygon Amoy Testnet, який слугує тестовим майданчиком для мережі Polygon PoS, та ThirdWeb SDK, що надає набір інструментів для створення додатків для децентралізованих платформ.

Вступ

Актуальність

Цифровий світ кардинально змінюється, і однією з найбільш захоплюючих інновай є поява невзаємозамінних токенів. Невзаємозамінні токени являються криптографічними токенами, підтверджені за допомогою технології блокчейн, що забезпечує їхню рідкісність.У міру розвитку технології блокечейн та її все ширшим розповсюдженням NFT, ймовірно, стануть поширеним способом безпечного та відкритого підтвердження права власності на різні види активів, як цифрових, так і фізичних. Транзакції з NFT можуть приносити значний дохід за рахунок комісійних, аукціонних продажів і роялті від вторинного продажу, тому при правильній стратегії та реалізації NFT-маркетплейс може стати прибутковим залучаючи колекціонерів, інвесторів та художників.

Мета курсової роботи це розробка децентралізованого веб-додатку, для купівлі, продажу та створення аукціонів для невзаємозамінних токенів для обміну та розповсюдження цифрового мистецтва. Ключовим аспектом розробки є забезпечення безпеки користувачів, мінімізувавши ризики витоку даних та втрати коштів.

Основне завдання проєкту - дослідження технології блокчейн, включаючи вивчення основних особливостей технології, виявлення інструментів, що допоможуть покращити застосунок та пришвидшити його розробку

Розділ 1
Технічні відомості

1.1 Еволюція від Web 1.0 до Web 3.0

Тім Бернес-Лі відіграв ключову роль на початкових етапах створення Інтернету в 1990 році оскільки створив три фундаментальні технології: HTML, URI, HTTP, які стали основою Інтернету включаючи найперший браузер веб-сторінок WorldWideWeb.app. Перед появою World Wide Web існувала програма Enquire, яка була простою гіпертекстовою системою з деякими ідеями, схожими на ті, що використовуються у Web та Semantic Web. Як описував Тім Бернес-Лі у своїй роботі “Information Management: A Proposal” : « In 1980, I wrote a program for keeping track of software with which I was involved in the PS control system. Called Enquire, it allowed one to store snippets of information, and to link related pieces together in any way. To find information, one progressed via the links from one sheet to another, rather like in the old computer game "adventure". I used this for my personal record of people and modules. It was similar to the application Hypercard produced more recently by Apple for the Macintosh. A difference was that Enquire, although lacking the fancy graphics, ran on a multiuser system, and allowed many people to access the same data.» [23]

В середині 1990-х років поява таких веб-браузерів, як Netscape Navigator, відкрила еру Web 1.0. Сайти надавали статичний контент, а не динамічні веб-сторінки на мові гіпертекстової розмітки. Дані та наповнення сайту надходили зі статичної файлової системи, а не з бази даних, а інтерактивність веб-сторінок була обмеженою. Web 1.0 означає зміну парадигми використання Інтернету.

На початку 21-го століття на зміну статичним веб-сторінкам прийшли інтерактивність, соціальні зв'язки та створений користувачами контент. Web 2.0 робить можливим перегляд користувацького контенту мільйонами людей по всьому світу практично миттєво. Додатки Web 2.0 включають YouTube, Facebook, Flickr, Instagram, Twitter та інші соціальні медіа-платформи, які компанії-розробники могли монетизувати. Такі веб-технології як HTML5, CSS3 та фреймворки Javascript, на кшталт ReactJs, AngularJs, VueJs дозволяють компаніям створювати нові концепції, дозволяючи користувачам робити більший внесок у соціальну мережу. Як наслідок, оскільки Web 2.0 створений навколо людей, розробникам просто необхідно створити систему для розширення можливостей і залучення користувачів. У Web 2.0 ви не можете контролювати свої дані або те, як вони зберігаються. Насправді корпорації регулярно відстежують і зберігають дані користувачів без їхнього відома чи згоди. Всі ці дані потім належать і управляються компаніями, які відповідають за ці платформи.

Web 3.0, також відомий як Семантичний Веб або читати-записувати-виконувати, - це етап (починаючи з 2010 року), який передбачає майбутнє Інтернету. Web 3.0 забезпечить децентралізований доступ до пов'язаних даних, на відміну від Web 2.0, де дані переважно зберігаються централізовано. Web 3.0 дозволить людям взаємодіяти з даними в поєднанні з технологіями штучного інтелекту і машинного навчання, об'єднуючи ідею Тіма Бернера Лі про семантичну павутину. Web 3.0 створить можливість для децентралізованих додатків замінити централізовані соціальні мережі, де користувачі зберігатимуть контроль над своїми даними. Додаток базується на блокчейні, що є розподіленою мережею і складається з багатьох однорангових вузлів або їхньої комбінації [2] [3]

1.2 Технологія блокчейн

Блокчейн - це технологія розподіленого реєстру (DLT), яка використовує мережевий консенсус для запису, виконання та перевірки транзакцій. Коли контроль централізований на одному суб'єкті, цілісність інформаційної системи безпосередньо залежить від цього суб'єкта, тоді як в децентралізованій системі цілісність інформації залежить від усіх учасників мережі, в якій працює система. Таким чином, для зміни запису недостатньо одностороннього волевиявлення одного суб'єкта. Кожен член мережі може перевіряти та підтверджувати транзакції безпосередньо, не віддаючи контроль над своїми даними третій стороні.[4]

Блокчейн можна описати як зростаючий стек записів, які називаються «блоками» і з'єднані між собою за допомогою криптографічних методів. Кожен блок повинен мати хеш-код попереднього блоку, мітку часу та набір підтверджених транзакцій. Будь-яка спроба підробити або змінити блок зробить недійсним весь ланцюжок, тому блокчейн є дуже стійким до шахрайства.[5]

1.2.1 Криптографія в блокчейні

Блокчейн був побудований на концепції забезпечення безпечного зв'язку між двома сторонами, тому він використовує криптографію для захисту транзакцій, які відбуваються між двома вузлами в мережі. Блокчейн використовує дві основні концепції - криптографію та хешування. Перша використовується для шифрування повідомлень в одноранговій мережі, а друга - для захисту інформації в блоці. Криптографічні алгоритми загалом можна розділити на два основних типи:

- Криптографія з симетричним ключем. Метод шифрування, при якому один і той самий ключ використовується як для шифрування, так і для

розшифрування даних. Основна ідея симетричного шифрування полягає в тому, що відправник та отримувач повинні мати однаковий ключ, використовуючи його для шифрування та розшифрування повідомлень. Існує два основних способи передачі спільного ключа: обмін через безпечний канал та використання протоколів обміну ключами. Однак виникають проблеми з безпекою, оскільки відправник і одержувач використовують спільний ключ і в разі його компрометації всі дані, що були зашифровані ключем можуть бути вкрадені. [6]

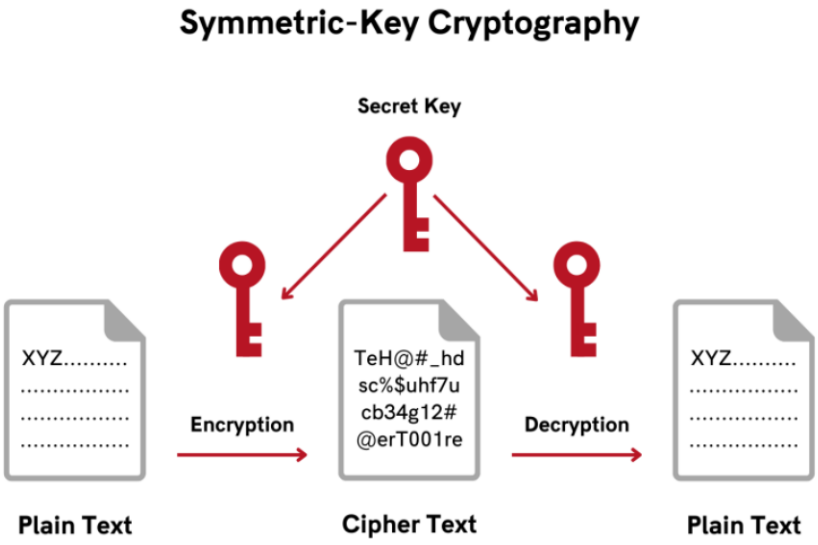

Рисунок 1 (Демонстрація роботи криптографії з симетричним ключем)

- Криптографія з асиметричним ключем. Кожен учасник мережі володіє парою криптографічних ключів - відкритим і закритим. Відкритий ключ є у вільному доступі і використовується для шифрування, в той час як приватний ключ залишається конфіденційним і використовується для розшифрування. Транзакції в блокчейні підписуються цифровим підписом закритим ключем відправника. Цей підпис слугує математичним доказом автентичності, гарантуючи, що транзакція походить від легітимного користувача. Для перевірки підпису

використовуються відкриті ключі всіх учасників, що сприяє зміцненню довіри та підзвітності в мережі. Кожен учасник мережі має доступ до відкритих ключів інших учасників, тому будь-який учасник може використати відкритий ключ відправника для перевірки підпису транзакції [8].

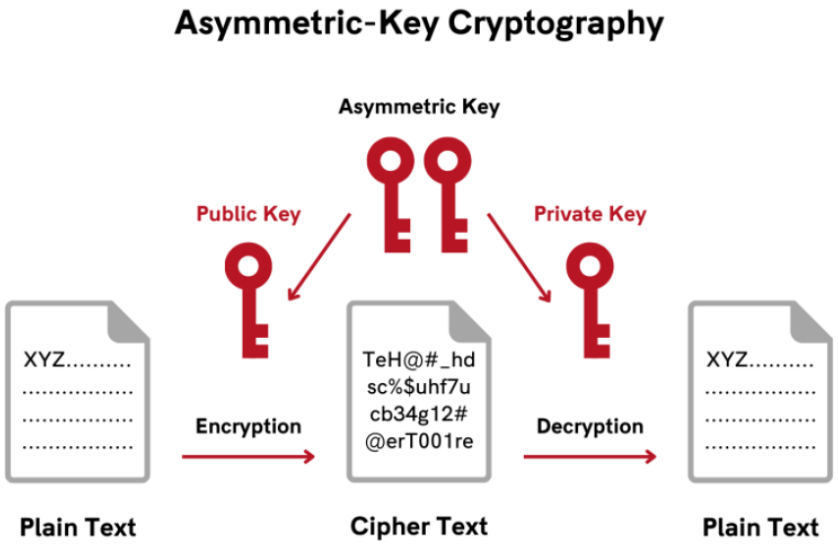

Рисунок 2 (Демонстрація роботи криптографії з асиметричним ключем)

1.2.2 Хешування та дерево Меркла

Хешування - це форма криптографії. Хеш-функція - це будь-яка функція, яка може зіставити дані довільної довжини зі значеннями фіксованого розміру. Значення, що повертаються хеш-функцією, називаються хеш-значеннями, хеш-кодами, дайджестами або просто хешами [7]. Цей метод використовує алгоритм для створення хеш-значення фіксованої довжини з відкритого тексту з відкритого тексту без використання ключів. Хеш-алгоритм перетворює будь-яку відкриту інформацію в унікальний рядок тексту. Незалежно від довжини вхідних даних, хеш завжди має наперед визначену довжину. Після завершення цього процесу він не може бути скасований, тобто оригінальний відкритий текст не може бути відновлений з хеш-значення. Навіть незначна зміна вхідних даних призведе до зовсім іншого хеш-значення на виході. Ця характеристика

полегшує швидке та ефективне виявлення загроз, підвищуючи безпеку методу шифрування.[6]

Блокчейн отримав свою назву через структуру, оскільки кожен блок даних з'єднаний з попереднім блоком у послідовний ланцюжок за допомогою хешу, утворюючи таку структуру даних як дерево Меркла. Дерево Меркла - це бінарне дерево, утворене хеш-покажчиками, і назване на честь його творця, Ральфа Меркла.

Дерево Меркла будується від рівня листових вузлів до рівня кореня Меркла шляхом групування вузлів попарно і обчислення хешу кожної пари вузлів на цьому рівні [9].

Корінь Меркла (Merkle root), також відомий як кореневий хеш - це найвищий хеш, що являється верхнім вузлом у дереві Меркла. Корінь Меркла формується шляхом об'єднання хеш-сум окремих блоків даних у дереві Меркла. При формуванні дерева кожен батьківський елемент генерується шляхом обчислення хеш-суми його дочірніх вузлів. Це висхідний тип побудови, де хеш-значення рухаються в напрямку знизу вгору. Отже, порівнюючи структуру дерева Меркла зі звичайною структурою даних бінарного дерева, можна помітити, що дерева Меркла насправді перевернуті донизу. [10] [9]

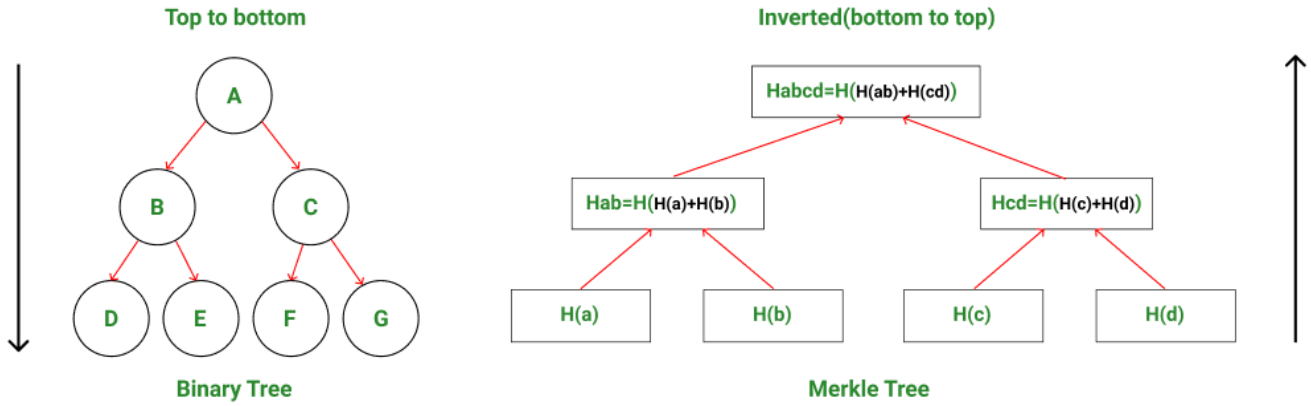

Рисунок 3 (Напрямок бінарного дерева проти напрямку дерева Меркла)

Така структура забезпечує швидкий спосіб перевірки цілісності даних і виявлення будь-яких змін в них. Якщо дані в одному з дочірніх вузлів були модифіковані, хеш батьківського вузла буде відрізнятися, що дозволяє виявити ці зміни. Дерева Меркла використовуються в блокчейні для забезпечення цілісності даних і

надання безпечного методу перевірки вмісту блоку. Вузол може перевірити дійсність всіх транзакцій, які зберігаються в блоці, гарантуючи, що в блокчейн не потрапляють шахрайські транзакції.

Підсумовуючи, дерева Меркла забезпечують безпеку : будь-яка недійсна чи випадково змінена інформація легко виявляється, а тому буде збережена цілісність блоків та транзакцій.

1.2.4 Механізм консенсусу

Механізм консенсусу – це алгоритм, за допомогою якого всі вузли мережі блокчейн досягають спільної згоди щодо поточного стану розподіленого реєстру. Таким чином, алгоритми консенсусу забезпечують надійність мережі блокчейн і встановлюють довіру між невідомими одноранговими вузлами в децентралізованому обчислювальному середовищі. Механізм консенсусу гарантує, що кожен новий блок, доданий до блокчейну, є істинним, дійсним і узгодженим усіма вузлами мережі блокчейн [4].

Одним з популярних механізмів консенсусу є Proof-of-Work (PoW), який використовується в блокчейнах Bitcoin та Ethereum. У PoW майнери змагаються у вирішенні складних математичних головоломок для підтвердження транзакцій і додавання нових блоків до ланцюжка. Основна ідея цього алгоритму полягає в тому, що вузли мережі повинні вирішити складну математичну головоломку і надати рішення, яке можна легко перевірити. Ця математична головоломка вимагає обчислювальних зусиль, тому вузли повинні знайти рішення головоломки якомога швидше, щоб створити новий блок і отримати за це винагороду.

Механізм консенсусу відіграє ключову роль у забезпеченні безпеки та цілісності даних

в мережі. Вибір механізму консенсусу залежить від таких факторів, як конкретні потреби програми, вимоги до масштабованості і бажаний рівень децентралізації.
Популярні механізми консенсусу :

- Proof-of-Work (PoW): вимагає від вузла-учасника довести, що виконана ним робота дає йому право на додавання нових транзакцій до блокчейну. Даний механізм безпечний, але він є обчислювально дорогим і енергоємним [11].
- Proof-of-Stake (Proof-of-Stake, PoS): виник як недорога і енергоефективна альтернатива алгоритму PoW. Він передбачає розподіл відповідальності за ведення публічного реєстру між вузлами-учасниками пропорційно до кількості наявних у них токенів віртуальної валюти. Однак цей алгоритм має той недолік, що він стимулює накопичення, а не витрачання.
- Proof-of-Authority (PoA): покладається на відомі та авторитетні валідатори для створення блоків і, таким чином, забезпечує обчислювальну потужність мережі. PoA більш швидкий і масштабований, ніж PoW, але менш децентралізований [12]

1.3 Огляд технології смарт-контрактів

Смарт-контракт, як і будь-який інший контракт, встановлює умови угоди. Один смарт-контракт може мати кілька різних умов, а один додаток може мати кілька різних смарт-контрактів для підтримки взаємопов'язаного набору процесів. Але на відміну від традиційного контракту, умови смарт-контракту виконуються у вигляді коду, що працює на блокчейні, такому як Ethereum. Крім того, смарт-контракт можна розуміти як комп'ютерний протокол, розроблений для полегшення, перевірки та примусового виконання договору або виконання контракту, що може бути виконаний або примусово виконаний самостійно при виконанні заздалегідь визначеної умови.[4]

Смарт-контракт відображає три ключові аспекти блокчейну, а саме: прозорості даних, захисту від підробки та постійної роботи. Інформація в блокчейні є вічною. Після того, як додаток зі смарт-контрактом додано до блокчейну, його, як правило, не можна скасувати або скоригувати.

Смарт-контракти - це обчислювальні інструкції, які формалізують елементи договірних відносин і можуть автоматично виконувати закодовані в них договори, як тільки виконуються заздалегідь визначені умови [4]. Працюючи на децентралізованому блокчейні замість централізованого сервера, смарт-контракти дозволяють кільком сторонам досягти спільного результату в точний, своєчасний і захищений від втручання спосіб, як це показано на рисунку 4. Смарт-контракти за своєю природою подібні до умовної конструкції в мовах програмування if-else: якщо X то Y [4]. Укладаючи угоду учасники визначають умови за яких можна буде виконати смарт контракт, якщо вони будуть виконані смарт-контракт автоматично здійснить транзакцію, що в подальшому закарбується в блокчейні.

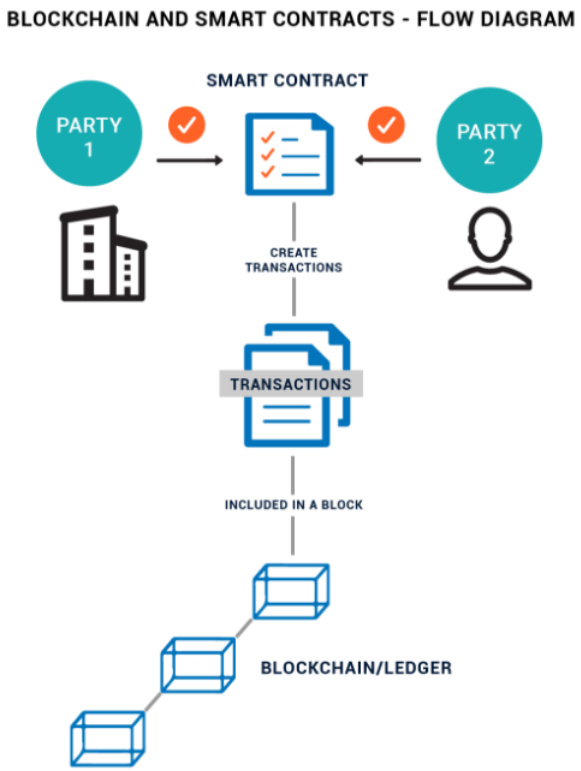

Рисунок 4 (Етапи укладання угоди з використанням смарт-контрактів)[13]

Розділ 2

Невзаємозамінні токени або NFT

За останні роки невзаємозамінні токени (NFT) привернули до себе багато уваги у світі цифрового мистецтва та суспільстві, але вони мають набагато давнішу історію, ніж більшість людей знає.

NFT було створено досить давно, але до недавнього часу воно не було таким популярним, як зараз. Як повідомляється, першим проданим NFT був "Quantum", розроблений і токенізований Кевіном МакКоєм у 2014 році на одному блокчейні (Namecoin), потім викарбуваний на Ethereum і проданий у 2021 році [2]. Однак, про Quantum забули після його випуску в 2014 році. Значною мірою це було пов'язано з

тим, що його початково викарбували в Namecoin, відгалуження біткоїна, яке існувало до Ефіріуму. Зокрема, Quantum жив у блоці Namecoin 174923, і саме там він залишався протягом років, до початку великого зростання інтересу до NFT у 2021 році.

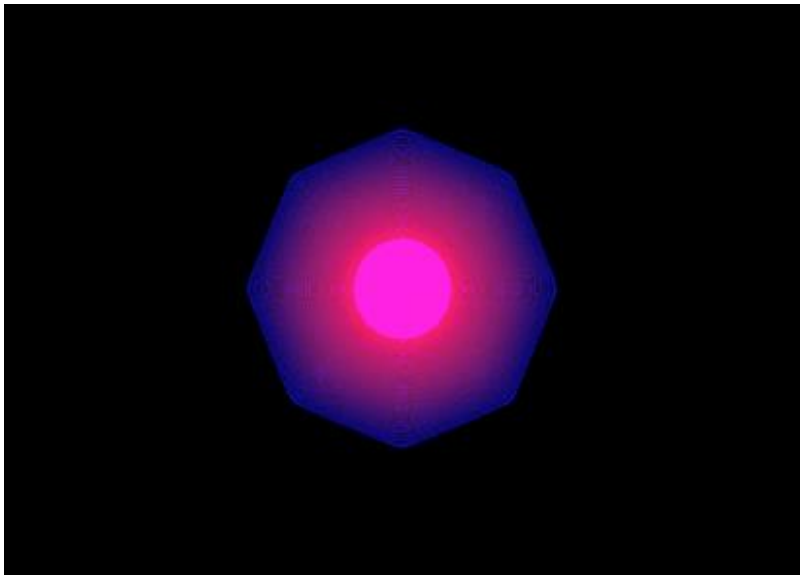

Рисунок 5 (Quantum. Автор: Kevin McCoy).

Для створення перших NFT використовувався смарт-контракт, відомий як стандарт ERC-721, який дозволяв створювати та володіти окремими цифровими активами, які можна було купувати, продавати та обмінювати на блокчейні.

Першими, хто почав використовувати NFT, були переважно розробники та шанувальники блокчейну, які побачили потенціал для створення унікальних і цінних цифрових активів на блокчейні. Створення цифрових артефактів, творів мистецтва та ігрових об'єктів, які можна було б купувати, продавати та обмінювати на блокчейні, було одним з перших випадків використання NFT.

2.1 Поняття та переваги NFT

Традиційно криптовалюти, такі як біткоїн, є взаємозамінними, тобто кожна одиниця BTC точно така ж, як і інша одиниця BTC, і їх можна обмінювати одна на одну без

будь-яких додаткових міркувань. Взаємозамінність є однією з фундаментальних властивостей традиційних валют, таких як долар США. У той же час, кожен невзаємозамінний токен має відмінний та унікальний ідентифікатор який відрізняє його від інших, що діє як доказ справжності та права власності у цифровому світі. NFT тобто Non-Fungible Token - це цифрові активи, такі як твори мистецтва, відео, музика тощо, які можна відстежувати за допомогою унікального ідентифікатора, що зберігається в блокчейні. Коли ви купуєте NFT, токен карбується для вас, і він діє як сертифікат автентичності та доказ права власності на цифровий актив, який він представляє.

Основна цінність NFT полягає в тому, що ,зберігаючи інформацію про власність у блокчейні, NFT вирішують питання походження цифрових творів мистецтва . У випадку зі звичайними цифровими файлами, він може бути скопійований і розповсюджений без можливості дізнатися, хто є його автором, або отримати будь-яке визнання чи цінність від своєї роботи. Файл також може бути видалений і втрачений назавжди.

NFT дозволяють простежити чіткий ланцюжок прав власності, а автори можуть ставити умови перепродажу NFT, щоб продовжувати отримувати вигоду, якщо цінність їхньої роботи зростає.

NFT являється новим способом володіння цифровими творами мистецтва та іншими цифровими активами, а технологія, що лежить в його основі, має потенціал революціонізувати не лише цифрове мистецтво та колекціонування, але й музичну індустрію та інші галузі, в яких права власності та походження мають важливе значення.

На відміну від криптовалют, кожен NFT має власну унікальну цінність і не може бути обміняний на аналогічний. Це означає, що коли ви купуєте NFT, ви купуєте щось унікальне, що не може бути безпосередньо обміняне на щось інше.

2.2 Технологія роботи NFT

NFT створюються за допомогою процесу, який називається карбуванням, в якому інформація про актив шифрується і записується в блокчейн. На високому рівні процес карбування передбачає створення нового блоку, перевірку інформації NFT валідатором і закриття блоку. Процес карбування часто передбачає включення смарт-контрактів, які визначають право власності та керують переказами NFT.

Під час карбування токенів їм присвоюється унікальний ідентифікатор, безпосередньо пов'язаний з однією адресою в блокчейні. Кожен токен має власника, і інформація про власника (тобто адреса, за якою знаходиться викарбуваний токен) є загальнодоступною. Навіть якщо буде викарбувано 5 000 NFT одного і того ж найменування, кожен токен має унікальний ідентифікатор і може бути розрізнений від інших.

Багато блокчейнів можуть створювати NFT, але вони можуть називатися по-різному. Наприклад, у блокчейні Bitcoin вони називаються Ordinals. Як і NFT на основі Ethereum, Bitcoin Ordinal можна купувати, продавати і обмінювати. Різниця полягає в тому, що Ethereum створює токени для активу, в той час як Bitcoin Ordinal мають серійні номери, так звані ідентифікатори, присвоєні сатоши - найменшому номіналу біткоїна.

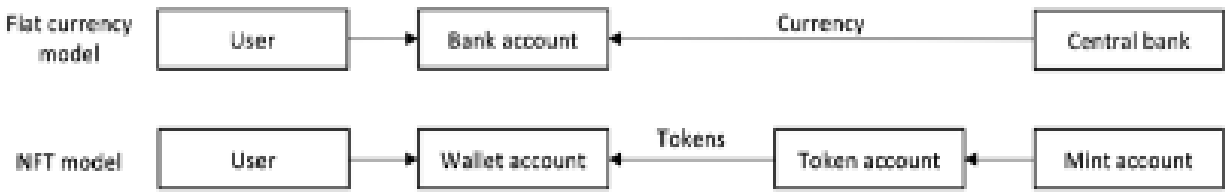

Рисунок 6 (Порівняння моделі традиційних валют і моделі NFT) [14]

На рисунку 6 показано аналогію між моделлю фіатної валюти та моделлю NFT. У моделі фіатної валюти центральний банк (наприклад, Європейський центральний банк, Федеральний резервний банк) відповідає за створення валюти. Валюта надається користувачам через їх банківські рахунки. На противагу технології NFT, обліковий запис карбує токени, а користувачі системи NFT можуть володіти токенами за допомогою облікових записів токенів. Точніше, користувачі екосистеми NFT взаємодіють зі своїм токен-рахунком через гаманець, який є програмним забезпеченням, пов'язаним з парою криптографічних ключів [14].

2.3 Метадані NFT

Метадані здебільшого розглядаються як дані, що надають інформацію про інші дані. Таким чином, традиційне визначення метаданих є дещо спеціалізованим у випадку NFT, оскільки цей термін включає як атрибути, так і саму оригінальну інформацію. Метадані невзаємозамінних токенів – це набір даних, що визначають їх зміст, і зазвичай представлені у форматі JSON, описуючи основні характеристики.

Зазвичай метадані визначаються автором NFT і це інформація яку він хоче помістити у свій майбутній токен, як-от назва, дата створення, власне ім’я, опис і тому подібне. Проте при використанні окремої платформи для генерації NFT, такі дані можуть бути створені автоматично.

Коли метадані існують в мережі , *on-chain metadata*, то вони вбудовуються безпосередньо в смарт-контракт. Це дає змогу скористатися основними перевагами сталості токенів, оскільки блокчейн є незамінним, а отже, включення цифрового активу в оригінальний смарт-контракт гарантує дефіцитність і тривале існування активу. Даний актив буде завжди існувати в тому вигляді, в якому його було створено, що і створює дефіцитність оскільки неможливо сформувати додаткові одиниці або змінити існуючі без консенсусу всіх учасників мережі [15].

У той час як поза мережеві метедані, *off-chain metadata*, знаходять поза блокчейном. Цифрові об'єкти такі як зображення, відео, аудіо або будь-які інші не можна просто зберігати безпосередньо на блокчейні Ethereum, оскільки вони можуть бути дуже об’ємними і їх зберігання безпосередньо на ланцюжку блоків було б дорого і неефективно. Стандарт ERC-721 був адаптований для включення посилання tokenURI, яка вказує смарт-контракт на публічну URL-адресу, що повертає словник даних у форматі JSON, який містить як посилання на цифровий

актив, як показано на рисунку 7 [15].

```
{
  "name": "Duke Khanplum",
  "image":
"https://storage.googleapis.com/ck-kitty-
image/0x06012c8cf97bead5deae237070f9587f8e7a2
66d/1500718.png",
  "description": "Heya. My name is Duke
Khanplum, but I've always believed I'm King
Henry VIII reincarnated."
}
```

Рисунок 7 (Приклад JSON-словника off-chain NFT, що містить метадані активу)

2.4 Смарт-контракти для NFT

Взаємодія смарт-контрактів і невзаємозамінні токени утворюють ефективний механізм для володіння і торгівлі цифровими активами: якщо NFT дозволяють створювати цифрове мистецтво, то смарт-контракти забезпечують функціональну структуру для їх створення та передачі. Смарт-контракти NFT пропонують децентралізовану структуру, яка не тільки забезпечує автентичність і походження цифрових активів, але й впроваджує інноваційні механізми для захисту транзакцій і оптимізації торгових процесів. Смарт-контракти автоматично виконують транзакції та операції, коли виконуються їхні умови, тим самим спрощує транзакції, усуваючи потребу в посередниках.

Використання смарт-контрактів для NFT [16]:

- Карбування. Коли було викарбувано новий невзаємозамінний токен, смарт-контракт визначає його використання та властивості, таким чином набуваючи юридичної сили та закріплюючи право власності за творцем.
- Торгівля. Під час купівлі або продажу NFT, дані про власника оновлюються в блокчейні, на якому існує смарт-контракт. Заносячи дані про власність у блокчейн-реєстр, смарт-контракти забезпечують захист від підробки та гарантують чітку, контрольовану і прозору історію володіння.
- Маркетплейси та роялті. Роялті за NFT – це механізм, що дозволяє творцям отримувати відсоток від вторинних продажів на маркетплейсах, що забезпечує потік доходу для авторів, що являється способом заробітку. Після створення цифрового мистецтва художник може укласти смарт-контракт з угодою про роялті на свій розсуд, що зазвичай складає 5-15% від ціни продажу. Коли NFT продалось, смарт-контракт автоматично виконується і надсилає суму роялті на цифровий гаманець автора.

  У деяких випадках маркетплейси використовують набір смарт-контрактів для продажу NFT на аукціонах. Ці майданчики можуть тимчасово утримувати право власності на токени до виконання заздалегідь визначених умов, таких як конкретна дата або ціна пропозиції. Вони також можуть встановлювати певні параметри для скасування аукціону і передачі права власності на NFT назад творцеві, якщо прийнятна ціна не буде досягнута.

2.5 Процес творення та розповсюдження NFT

Зацікавлені у володінні NFT можуть отримати його кількома різними способами. Перший спосіб це створити свій власний токен, тобто викарбувати новий і даний процес називається «minting».

У більшості систем блокчейн NFT створюються через взаємодію зі смарт-контрактом. Існує багато шаблонів смарт-контрактів для створення NFT, доступних у різних відкритих джерелах, зокрема на блокчейн-платформах, які підтримують дані токени, у відомих творців цієї галузі й на ринках NFT.

Невзаємозамінні токени створюються та реєструються в блокчейні за допомогою спеціального процесу. Майже будь-який фрагмент даних – від одного рядка тексту до цілого світу віртуальної реальності – можна закарбувати у формі NFT. Після створення токену відповідний цифровий медіа-файл часто завантажується на зовнішній ресурс . Смарт-контракт розгортається в мережі блокчейн, накшталт Ethereum, визначаючи правила і функціх для створення, володіння і передачі NFT. Творець використовує функції смарт-контракту, спеціально розроблені для карбування, щоб конвертувати всі активи в токени і на даний момент є суб’єктом карбування.

Автор NFT пов’язує інформацію про актив зі смарт-контрактом і встановлює умови, такі як ціна і максимальну кількість на адресу, щоб інші особи могли карбувати NFT у майбутньому. Користувачі, які хочуть створити NFT, взаємодіють зі смарт-контрактом за допомогою виклику функції карбування, що зазвичай здійснюється через веб-інтерфейс або децентралізований додаток, підключений до блокчейну. Таким чином створюється токен для активу і особа отримує право власності на актив.

Для карбування мають бути виконані умови, визначені в смарт-контракті, що користувач повинен заплатити певну вартість для карбування. З цієї причини карбування також означає придбання нещодавно запущеного токену. В цей час суб'єкт, що карбує NFT, стає його покупцем.[1]

Смарт-контракти, що створюють NFT, вимагають сплати комісії, так звані плати за газ, учасникам мережі, які називаються валідаторами й забезпечують достовірність стану володіння NFT. Газові збори - це платежі, які здійснюються користувачами для компенсації учасникам мережі, яких часто називають майнерами або валідаторами, за обчислювальні ресурси, необхідні для обробки та перевірки транзакцій в мережі блокчейн. Валідатори виділяють обчислювальні потужності для перевірки транзакцій і підтримки цілісності мережі блокчейн. Тому плата за газ слугує стимулом для валідаторів продовжувати виконувати ці завдання чесно та ефективно, отримуючи компенсацію у вигляді газових зборів.

Другий спосіб заволодіти NFT це придбати вже раніше створений екземпляр на одному із існуючих маркетплейсів. У цьому випадку в транзакції можуть брати участь кілька сторін, наприклад, продавець отримує ціну за свій токен, а блокчейн - комісію за транзакцію (газовий збір). Якщо в транзакції бере участь посередник, наприклад, маркетплейс, то він отримує комісію, продавець - ціну за свій токен, оригінальний творець, якщо він відрізняється від продавця, - роялті від перепродажу, і, нарешті, блокчейн - комісію за транзакцію.

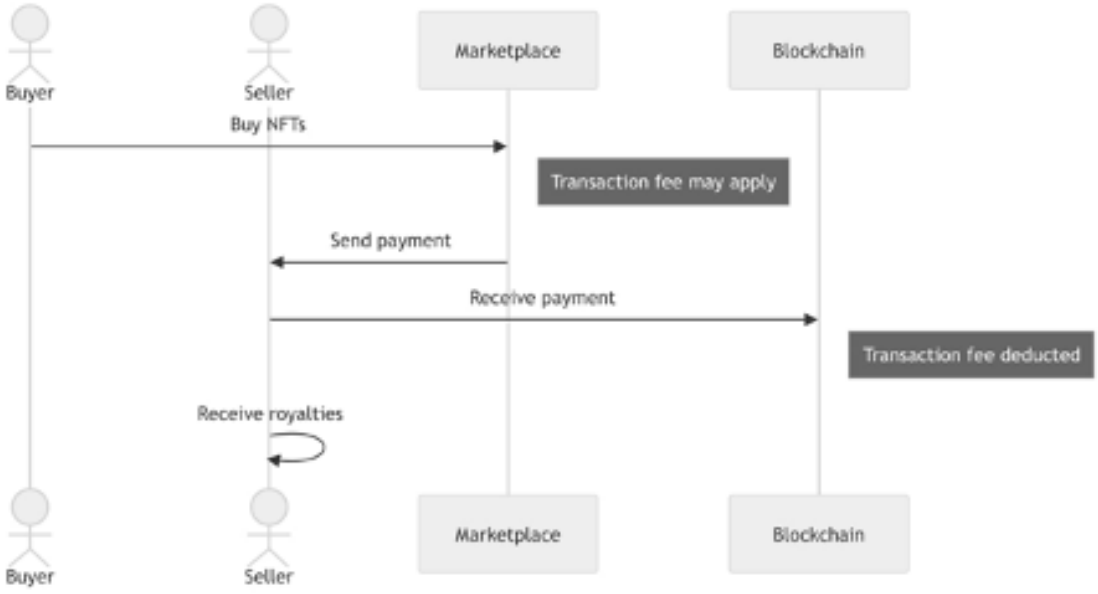

Рисунок 8 (Демонстрація роботи маркеплейсу)

Загальний процес створення та розповсюдження проектів NFT можна умовно поділити на 3 етапи, як показано на рисунку 9 [1]:

- Етап перший. Створення необхідних файлів для токена, включаючи сам актив і файли метаданих JSON. Всі ці файли зберігаються поза ланцюгом, тобто не зберігаються безпосередньо в блокчейні. Крім того, метадані повинні також містити місцезнаходження реального активу, тому JSON файл створюється після створення та збереження цифрового мистецького об’єкту.
- Етап другий. Реалізація смарт-контракту NFT і розгортання транзакції для створення контракту в мережі блокчейн. Ця транзакція містить байт-код смарт-контракту після чого, при включенні транзакції в блок, генерується адреса контракту, яка використовується як ідентифікатор смарт-контракту NFT. Якщо смарт-контракт NFT належним чином розгорнуто в блокчейні, URI метаданих, створений на попередньому етапі, встановлюється в смарт-контракті NFT.
- Етап третій. Карбування NFT, що відбувається за допомогою смарт-контракт і може бути здійснено кількома способами. Творець NFT безпосередньо карбує всі NFT і розміщує їх на NFT-маркетплейсі. Інший метод полягає в тому, що користувачі платять ціну за карбування NFT протягом певного періоду, який називається падінням NFT, доки загальна кількість випущених NFT не зрівняється із загальною пропозицією. Також існує спосіб карбування при якому користувачі карбують токени під час

періоду падіння, але без сплати комісії.

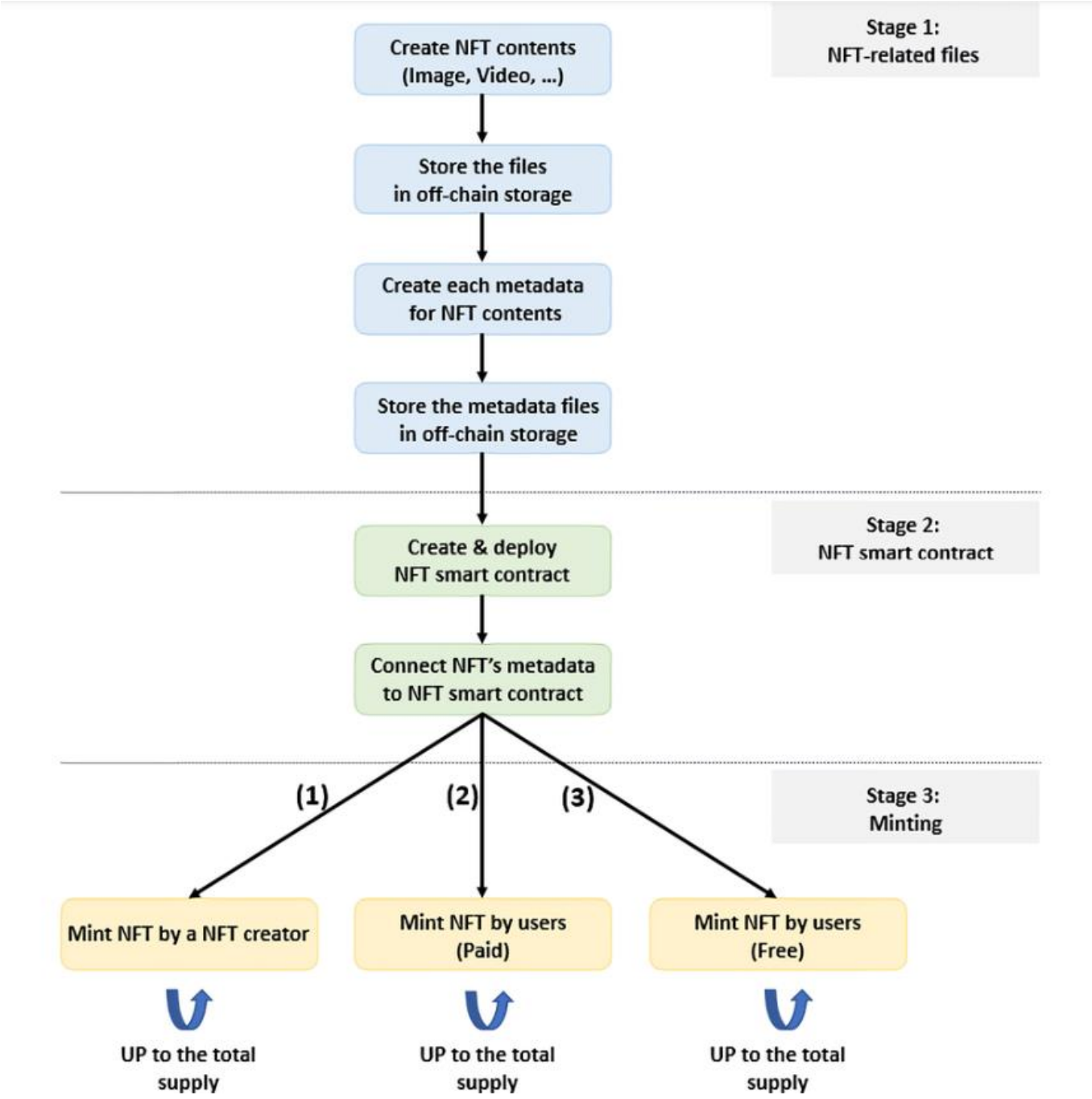

Рисунок 9 (Загальний процес створення NFT проектів)

Розділ 3

Деталі розробки системи

3.1 Технічне завдання

Розробити застосунок який буде слугувати як платформа для безпечної купівлі та продажу невзаємозамінних токенів. Надати також користувачам можливість участі в аукціонах, де вони зможуть визначати розмір ставки на власний розсуд і конкурувати за придбання потрібних активів. Застосунок повинен мати зручний та простий інтерфейс, що буде зрозумілим для користувача.

Мета розробки маркетплейсу: створити платформу для продажу та аукціону токенів на основі власної NFT колекції. Такий застосунок дозволить художникам та майстрам глобально представити свої картини, продавати їх безпосередньо покупцям, контролювати ціну та права на свої твори та отримувати пасивний дохід від подальших перепродаж.

3.1.1 Опис структури системи

Користувач може зареєструватися або авторизуватися на NFT маркетплейс використовуючи Metamask гаманець. Така система ідентифікації дозволяє зробити додаток більш безпечним, оскільки Metamask являється надійним інструментом, таким чином не будуть створені шахрайські облікові записи. Після успішного входу в систему користувачі може переміщатися застосунком для пошуку бажаних NFT. Якщо користувач знаходить екземпляр який йому сподобався він може купити право власності на цей токен за ціною визначеною поточним власником. NFT також можуть бути виставлені на розіграш в аукціоні, тому у

такому випадку його можна купити за поточною ставкою зробити свою. Власник, виставляючи NFT на аукціон, повинен зазначити термін дії аукціону, початкову ставку та ціну викупу. Угода буде здійснена за допомогою доступної цифрової валюти, яка зберігається в гаманці користувача. Коли підтвердиться транзакція покупки, на маркетплейсі буде відображатися адреса покупця, як нового власника. Придбані NFT користувач може переглянути в своєму особистому профілі або під час виставлення їх на продаж. Під час продажу токену потрібно вказати дату початку і закінчення періоду торгівлі (Listing start date/Listing end date), а також вартість.

3.1.2 Учасники системи

Маркетплейс платформа повинна демонструвати виставлені на продаж художні об’єкти навіть для тих користувачів, які не маю власного гаманця і не зареєстровані як застосунку. Система має гарантувати такі ролі учасників як : зареєстрований користувач та гість, користувач без прив’язаного гаманця.

3.1.3 Функціональні можливості системи для користувачів

Функціональність системи для зареєстрованого користувача:

Підв’язавши свій гаманець користувач може переглядати NFT, які виставлені на продаж іншими учасниками, купити їх або прийняти участь в аукціоні та перевірити які токени у нього наявні. Крім того, для нього є можливість виставити на продаж раніше куплені NFT.

Функціональність системи для гостя:

Даний користувач може переглядати NFT, що виставлені на продаж іншими учасниками. Також має можливість переглянути які токени розігруються на

аукціоні та яка мінімальна ставка. Для торгування NFT або здійснення покупки потрібно прив’язати Metamask гаманець.

3.2 Огляд інструментів розробки

- Visual Studio Code - Visual Studio Code поєднує в собі простоту редактора вихідного коду з потужними інструментами розробника завдяки розширенням, підтримує macOS, Linux та Windows тому є доступним для користувачів на різних платформах, що забезпечує широку аудиторію розробників. У Visual Studio Code доступно багато розширень для роботи з блокчейнами і криптовалютами, зокрема для Ethereum, які полегшують розробку smart-контрактів та взаємодію з Ethereum мережею [17]
- ThirdWeb – платформа для розробки, яка дозволяє легко створювати, запускати та керувати Web 3 додатками та іграми на будь-якому EVM-сумісному блокчейні. ThirdWeb пропонує інструменти та підтримку для децентралізованої розробки, надаючи уніфікований thirdweb SDK, що є потужним і гнучким набором інструментів для створення програмного забезпечення для dApps [18]
- Metamask - провідний гаманець для особистого контролю коштів, що пропонує безпечний та простий спосіб доступу та взаємодії до блокчейн-додатків. Metamask робить акцент на інтеграції з браузером та мобільними додатками, тому він надає зручний доступ до dApps та підтримує широкий спектр токенів та NFT на основі Ethereum. Генеруючи паролі та ключі на пристрої користувача, гарантує повну безпеку та контроль фінансів і токенів, оскільки ніхто не матиме доступ до цих даних [19].

- Next.js - це фреймворк React для створення full-stack веб-додатків. React-компоненти використовуються для створення користувацьких інтерфейсів, а Next.js надає додаткові функції та оптимізації у вигляді набору функціональних модулів і можливостей для розробки веб-додатків.
- Polygon Amoy Testnet - слугує тестовим майданчиком для мережі Polygon PoS і може взаємодіяти з Sepolia Testnet від Ethereum. Це дозволяє розробникам розгортати, тестувати і виконувати свої dApps в безризиковому і безкоштовному блокчейн-середовищі. Завдяки сумісності EVM Polygon, Amoy пропонує простий спосіб для розробників, які бажають перенести свої dApps з Ethereum на Polygon, що дозволяє їм проводити тестування перед розгортанням своїх dApps в основній мережі [20].
- ERC-721 - відноситься до стандарту невзаємозамінних токенів, який можна знайти на блокчейні Ethereum. Він надає набір рекомендацій для створення унікальних токенів, які представляють цифрові активи. Стандарт ERC-721 складається з набору функцій, які розробники можуть впроваджувати в свої смарт-контракти що може полегшити створення, передачі та управління NFT [2].
- IPFS - протокол, який підтримує адресацію вмісту, щоб користувачі могли знайти дані, якщо вони зберігаються на будь-якому вузлі мережі IPFS і являється найпоширенішим децентралізованим сховищем. IPFS надає можливість зберігати файли децентралізовано, не в блокчейні, що називають "поза мережевим" зберіганням [2].
- Pixlr - редактор для фото, який надає інструменти для створення зображень з використанням штучного інтелекту : Text to Image спосіб перетворення простого тексту на візуально захопливий витвір мистецтва.

3.3 Створення власної NFT колекції

Для того, аби наповнити маркетплейс токенами для продажу необхідно створити власну NFT колекцію. Малюнки для колекції було створено за допомогою Pixlr . Створюючи контракт за допомогою Thirdweb потрібно визначити адресу на яку має надійти виручка від первинного продажу та на яку має надходити дохід від роялті. Також потрібно заповнити метадані контракту : фото, назва та опис для колекції.

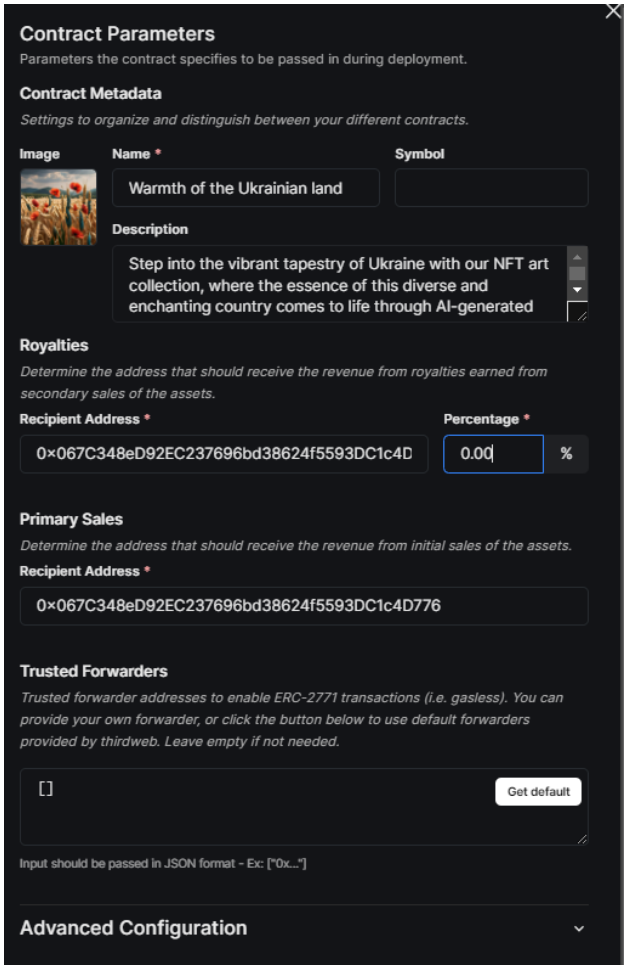

Рисунок 10 (Створення смарт-контракту для NFT колекції)

Фото для подальшої колекції було завантажено на Filebase - провідна платформа розробки, що надає користувачам швидкий доступ до сховища IPFS, виділених IPFS-шлюзів та IPNS-імен. На рисунку 11 демонструється приклад CSV-файлу з метаданими кожного NFT, які включають IPFS CID, де віддалено зберігається малюнок активу.

name	description	image
Kyiv Elegance	Image that encapsulates the essence of Ukraine's capital, Kyiv, striking a balance between its rich cultural heritage and modernity	https://ipfs.filebase.io/ipfs/QmcmGZNUUFEzCUYNaLpFVEhu1ViywgotjK3ZPGwXVgYf7R/Kyiv.jpg
Dnipro Harmony	Whether you seek solace in the quiet serenity of the wetlands or marvel at the intricate web of life thriving within, this NFT offers a glimpse into the timeless allure of the Dnipro Delta.	https://ipfs.filebase.io/ipfs/QmcmGZNUUFEzCUYNaLpFVEhu1ViywgotjK3ZPGwXVgYf7R/dnipro.jpg
Carpathian Mountains	The Carpathians demonstrate the power and strength of Ukraine	https://ipfs.filebase.io/ipfs/QmcmGZNUUFEzCUYNaLpFVEhu1ViywgotjK3ZPGwXVgYf7R/mountains.png
Cozy Ukrainian Village	The tranquil atmosphere of the countryside, with soft sunlight penetrating through the trees, casting spotted shadows on the cobbled streets.	https://ipfs.filebase.io/ipfs/QmcmGZNUUFEzCUYNaLpFVEhu1ViywgotjK3ZPGwXVgYf7R/village.png
Golden wheat fields	Captures the essence of Ukraine's rural beauty, focusing solely on the mesmerising sight of endless fields of golden wheat gently swaying in the breeze.	https://ipfs.filebase.io/ipfs/QmcmGZNUUFEzCUYNaLpFVEhu1ViywgotjK3ZPGwXVgYf7R/wheat.png

Рисунок 11 (Демонстрація CSV-файлу з метаданими NFT)

Даний CSV-файл завантажується в проект на Thirdweb і автоматично створюється колекція токенів, як показано на рисунку 12.

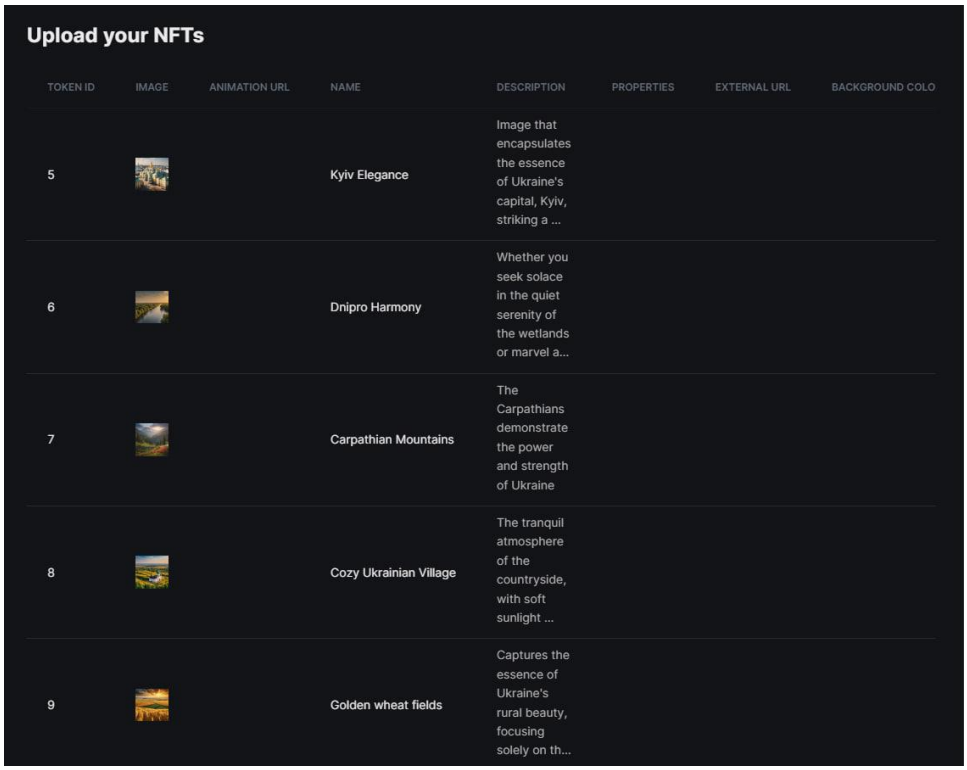

Рисунок 12 (Створення NFT на основі CSV-файлу)

Далі відбувається транзакція за розгортання смарт-контракту.

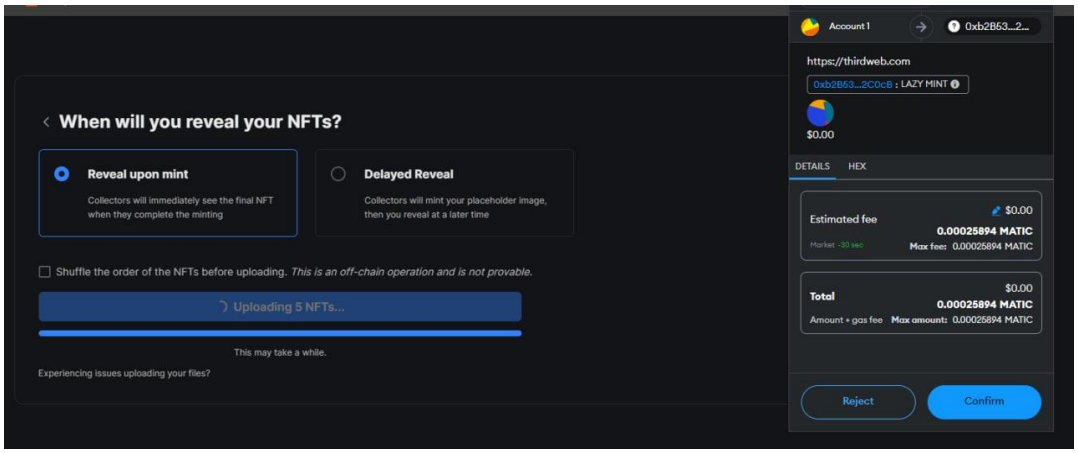

Рисунок 13(Підтвердження транзакції у Metamask)

Останнім кроком являється створення умов або claim conditions, що дозволять налаштувати різні аспекти дропу: хто може претендувати на токени, скільки

токенів буде випущено, ціна кожного токена, дата випуску і т.д, як показано на рисунку 14.

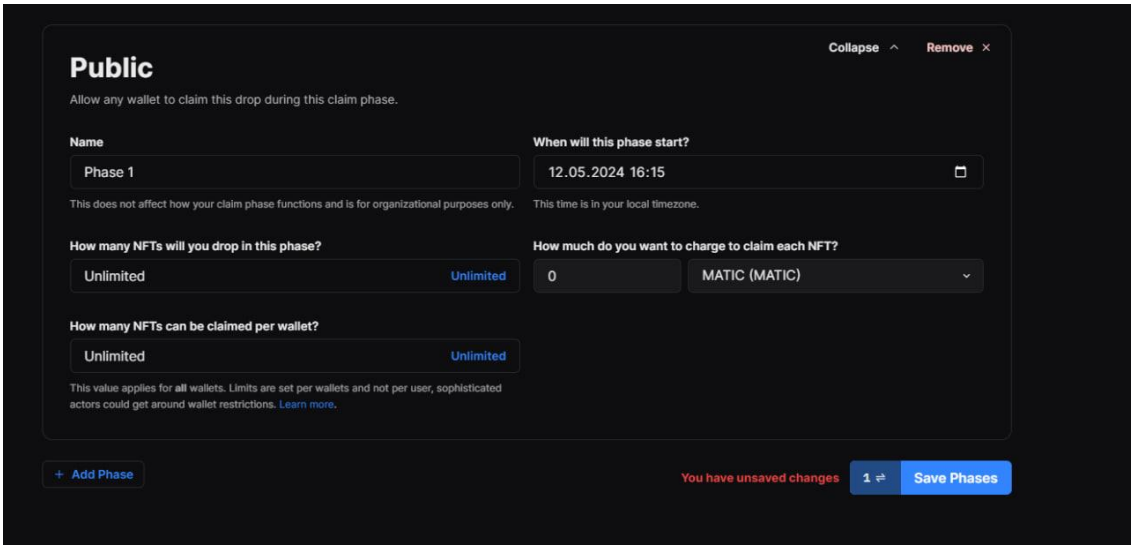

Рисунок 14 (Claim conditions)

3.4 Опис використанної React SDK

React SDK - це набір для розробки програмного забезпечення, що надається компанією thirdweb та призначений для використання з React. Даний інструмент дозволяє розробникам інтегрувати функціональність та можливості thirdweb у свої web3-застосунки для будь-якого EVM-сумісного бокчейну. React SDK є частиною великого пакету thirdweb SDK, який включає додаткові компоненти для різних платформ та функцій [21].

- *ConnectWallet* - компонент, який рендерить кнопку, яка при натисканні відкриває модальне вікно, що дозволяє користувачам підключатися до різних гаманців [22]. У застосунку використовується для підключення гаманця.

```
<ConnectWallet btnTitle="Login"/>
{address && (
  <NextLink href={`/profile/${address}`} passHref>
    <Link ml={"20px"}>
      <Avatar src='https://bit.ly/broken-link'/>
    </Link>
  </NextLink>
)}
```

Рисунок 15(Використання ConnectWallet в застосунку)

- *ThirdwebNftMedia* - компонент, який генерує NFT на основі заданого об’єкта метаданих. Якщо властивість зображення в метаданих є URL/IPFS URI, вона завантажується з джерела [22]. У застосунку використовується для відображення NFT на основі заданих метаданих

```
<Box borderRadius="4px" overflow="hidden">
  <ThirdwebNftMedia metadata={nft.metadata} height="100%" width="100%" />
</Box>
```

Рисунок 16(Використання ThirdwebNftMedia в застосунку)

- *Web3Button* - кнопка, яка при натисканні виконує функцію на смарт-контракті з підключеного гаманця. Перед тим, як викликати функцію контракту, вона перевіряє дотримання наступних критеріїв: існування підключеного гаманця та правильність мережі в якій знаходиться гаманець. У застосунку використовується для виставлення NFT на продаж, покупки NFT на аукціоні та виставлення власної ставки.

```
<Web3Button
  contractAddress={MARKETPLACE_ADDRESS}
  action={async () : Promise<void>  => {
    await handleSubmitDirect(handleSubmissionDirect)();
  }}
  onSuccess={(transactionResult : void ) : void  => {
    router.push( url: `/token/${NFT_COLLECTION_ADDRESS}/${nft.metadata.id}`);
  }}
>
  Create Direct Listing
</Web3Button>
```

Рисунок 17(Використання Web3Button в застосунку)

```
<Web3Button
  contractAddress={MARKETPLACE_ADDRESS}
  action={async () : Promise<any>  => buyToken()}
  isDisabled={(!auctionListing || !auctionListing[0]) && (!directListing || !directListing[0])}
>Buy at asking price</Web3Button>
<Text align={"center"}>OR</Text>
<Flex direction={"column"}>
  <Input
    mb={5}
    defaultValue={
      auctionListing?.[0]?.minimumBidCurrencyValue?.displayValue || 0
    }
    type={"number"}
    onChange={(e : React.ChangeEvent<HTMLInputEle... ) => setBidValue(parseInt(e.target.value))}
  />
  <Web3Button
    contractAddress={MARKETPLACE_ADDRESS}
    action={async () : Promise<any>  => await submitBidOffer()}
    isDisabled={!auctionListing || !auctionListing[0]}
  >Set Your Bid</Web3Button>
```

Рисунок 18 (Використання Web3Button в застосунку)

- *ThirdwebProvider* - компонент-обгортка, який надає доступ до всіх інструментів та компонентів інтерфейсу користувача SDK, інтегруючи можливості SDK, що дозволяють працювати з блокчейном [22]. У

застосунку використовується для використовується для налаштування SDK та забезпечення його функціональності.

```
<ThirdwebProvider activeChain={NETWORK} clientId={"1c208df539001fc95bd1ce76073f9ddd"}>
  <ChakraProvider>
    <Navbar/>
    <Component {...pageProps} />
  </ChakraProvider>
</ThirdwebProvider>
```

Рисунок 19(Використання Web3Button в застосунку)

- MediaRenderer - компонент, який рендерить будь-який ресурс, що зберігається на IPFS, за IPFS URI / URL. Актив завантажується з IPFS через IPFS-шлюз thirdweb [22]. У застосунку використовується для відображення медіа, яке знаходиться віддалено, на IPFS.

```
<Box borderRadius={"4px"} overflow={"hidden"} mr={"20px"}>
  <MediaRenderer src={contractMetadata.image} height="50px" width="50px" />
</Box>
```

Рисунок 20(Використання MediaRenderer в застосунку)

- useAddress - інструмент для отримання адреси підключеного гаманця, відображаючи поточну адресу користувача, яка пов'язана з його гаманцем, після успішного підключення до додатку або сайту [22]. У застосунку використовується для отримання адреси гаманця та використання її в подальших операціях

```
const address : string | undefined  = useAddress();
```

Рисунок 21(Використання useAddress в застосунку)

- useContract – інструмент для підключення до смарт-контракту. Надаючи адресу смарт-контракту першим параметром, після підключення змінна стане екземпляром смарт-контракту до якого підключається [22]. У застосунку використовується для отримання доступу до функцій та даних смарт-контракту, використовуючи його адресу.

```
const { contract: marketplace : MarketplaceV3 | undefined } = useContract(MARKETPLACE_ADDRESS, _contractType: "marketplace-v3");
const { contract: nftCollection : SmartContract<BaseContract> | ... } = useContract(NFT_COLLECTION_ADDRESS);
```

Рисунок 22(Використання useContract в застосунку)

- useValidDirectListings/ useValidEnglishAuctions - інструменти для отримання списку всіх дійсних списків продажів та аукціонів з контракту Marketplace V3 [22]. У застосунку вони використовуються для отримання цих списків з контракту Marketplace V3 для певного NFT.

```
const { data: directListing : DirectListingV3[] | undefined , isLoading: loadingDirectListing : boolean } =
  useValidDirectListings(marketplace, filter: {
    tokenContract: NFT_COLLECTION_ADDRESS,
    tokenId: nft.metadata.id,
  });

const { data: auctionListing : EnglishAuction[] | undefined , isLoading: loadingAuction : boolean } =
  useValidEnglishAuctions(marketplace, filter: {
    tokenContract: NFT_COLLECTION_ADDRESS,
    tokenId: nft.metadata.id,
  });
```

Рисунок 23(Використання useContract в застосунку)

- useCreateDirectListing/useCreateAuctionListing - інструменти для виставлення NFT на продаж або аукціон. UseCreateDirectListing використовується для створення прямого лістингу, коли продавець безпосередньо передає NFT у контракт маркетплейсу для продажу. UseCreateAuctionListing, натомість, дозволяє створювати аукціонні лістинги, де NFT залишаються у депонуванні протягом аукціону, і продавець має передати їх у контракт на маркетплейсі

під час створення лістингу [22]. У застосунку вони використовуються для створення нових лістингів на маркетплейсі для продажу NFT.

```
const { mutateAsync: createDirectListing } = useCreateDirectListing(marketplace);
const { mutateAsync: createAuctionListing } = useCreateAuctionListing(marketplace);
```

Рисунок 24(Використання useContract в застосунку)

- useOwnedNFTs - доступу до списку NFT, що належать одній адресі гаманця. Доступний для використання в смарт-контрактах, що реалізують розширення ERC721Enumerable, ERC1155Enumerable або ERC721Supply [22]. У застосунку використовується для отримання списку NFT, які належать поточному користувачеві та знаходяться у вказаному контракті NFT колекції.

```
const { data : NFTType[] | undefined , isLoading : boolean  } = useOwnedNFTs(contract, address);
```

Рисунок 25 (Використання useOwnedNFTs в застосунку)

- useNFTs - інструмент для запиту всіх NFT, пов'язаних зі смарт-контрактом. Доступний для використання в смарт-контрактах, що реалізують стандарт ERC721 або ERC1155 [22]. У застосунку використовується для отримання списку всіх NFT, які зберігаються в смарт-контракті, і використовуються для подальшого відображення, взаємодії або обробки даних.

```
const { data : NFT[] | undefined , isLoading : boolean  } = useNFTs(contract);
```

Рисунок 26 (Використання useNFTs в застосунку)

3.5 Демонстрація застосунку

На головну сторінку користувачі потрапляють при першому вході до застосунку. Дана сторінка слугує початковим екраном де коротко описана функціональність

системи. У верхній частині екрана розташована навігаційна панель, яка надає користувачам можливість переміщатися застосунком. Для повернення на головну сторінку користувач може натиснути на назву застосунку “NFTUniverse” та зможе повернутися на головний екран. Кнопка “Login” яка дозволяє авторизуватися за допомогою гаманця Metamask, щоб в подальшому користуватися усіма можливостями системи.

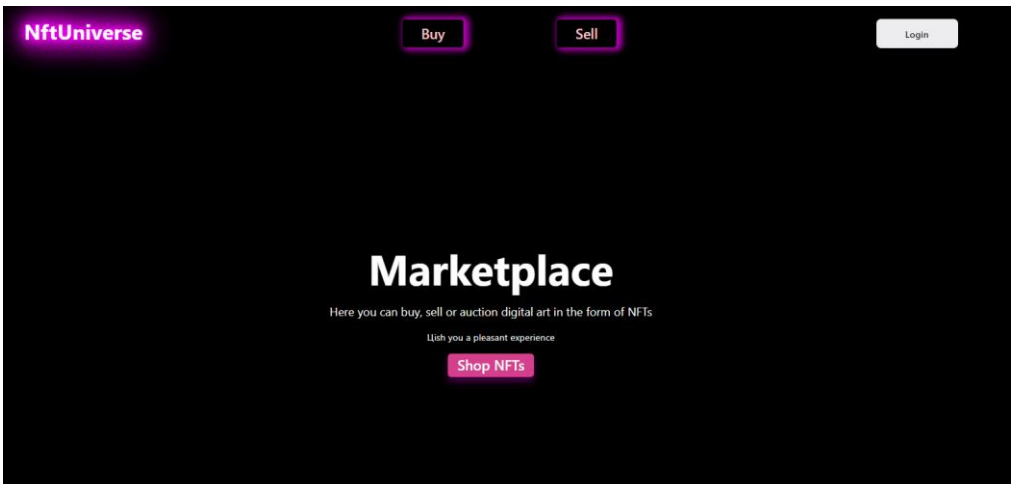

Рисунок 27 (Головна сторінка маркетплейсу)

Підключення до гаманця здійснюється через спливаюче вікно після натискання кнопки “Login”, як показано на рисунку 27. Користувач повинен обрати акаунт, за

допомогою якого бажає виконати вхід.

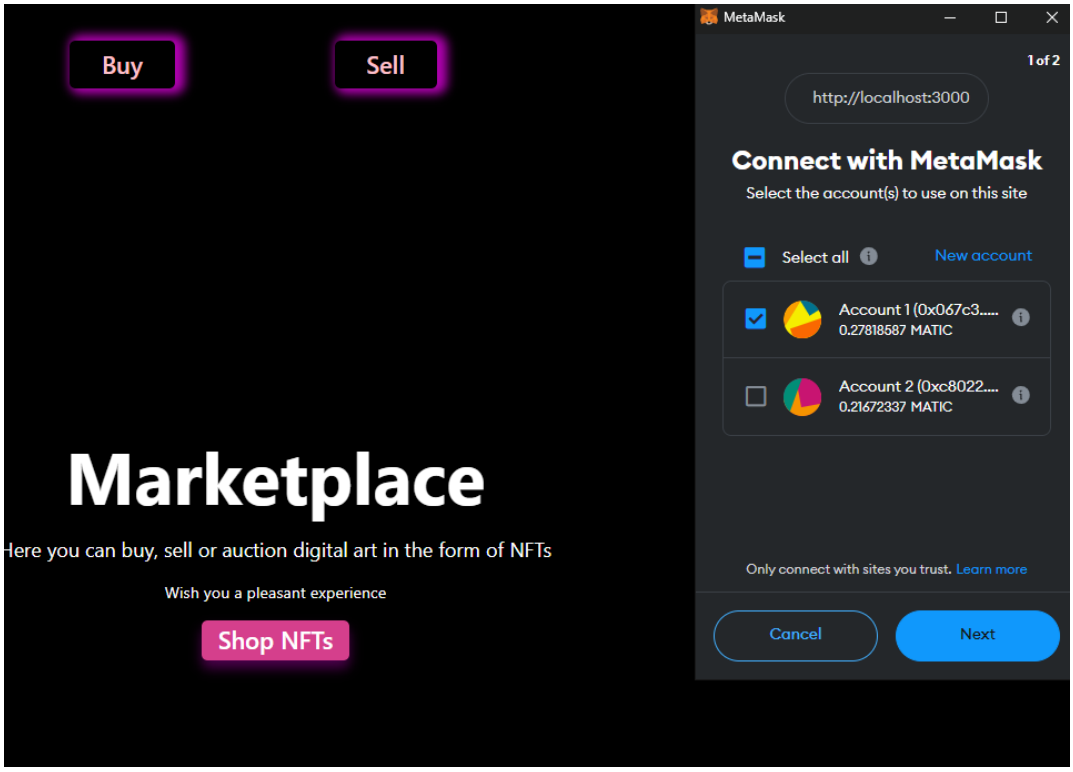

Рисунок 28 (Спливаюче вікно підключення гаманця Metamask)

Авторизувавшись, сторінка маркетплейсу буде оновлена автоматично, а гаманець залишиться підключеним до системи, як показано на рисунку 29, доки користувач вийде з застосунку.

Рисунок 29(Підключений гаманець Metamask до застосунку)

Щоб почати користуватися функціоналом платформи, а саме покупка токенів, можна скористатися кнопками “Shop NFTs” або “Buy”. У розділі продажу перед

користувачами постає NFT колекція, залежно від статусу кожного токена можна здійснювати покупки. Токен може бути виставлений на продаж, і відображатиметься ціна, встановлена продавцем. Наприклад, як показано на рисунку 30, токен “Golden Wheat Fields” коштує 0.002 MATIC. Якщо токен не може бути виставлений на продаж, замість ціни користувачі побачать текст "Not Listed", це демонструє токен “Dnipro Harmony”, що показано на рисунку 30. Також, на платформі може здійснюватися аукціон, тоді користувачі будуть бачити яка мінімальна ставка, як можна бачити на рисунку 30 на прикладі токену “Cozy Ukranian Village”.

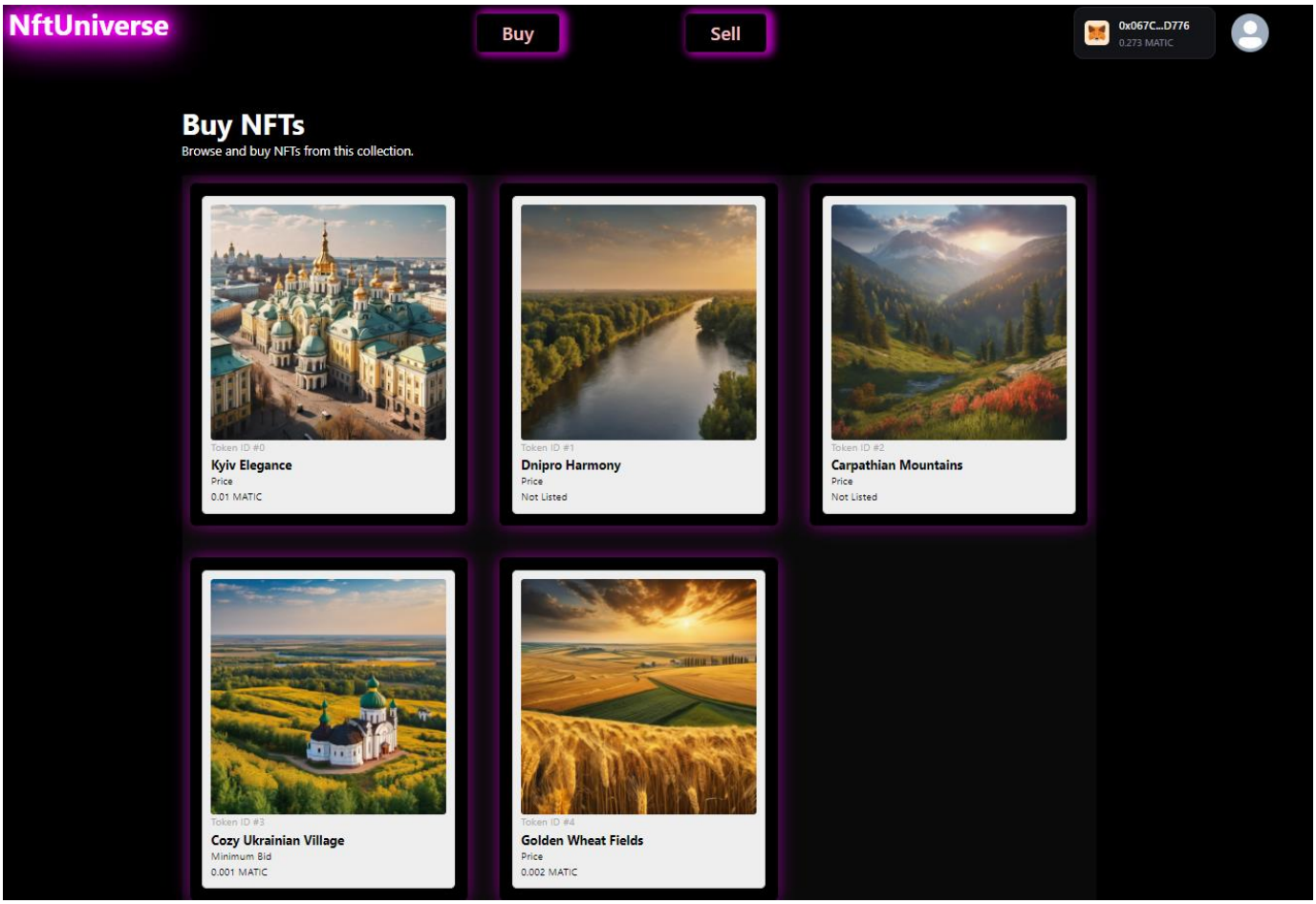

Рисунок 30 (Відображення сторінки купівлі NFT)

Виставлення NFT на продаж відбувається шляхом визначення терміну, під час якого він буде тривати та ціни, заплативши яку, користувач отримає токен. Після заповнення всіх полів з’являється спливаюче вікно для підтвердження транзакції через гаманець.

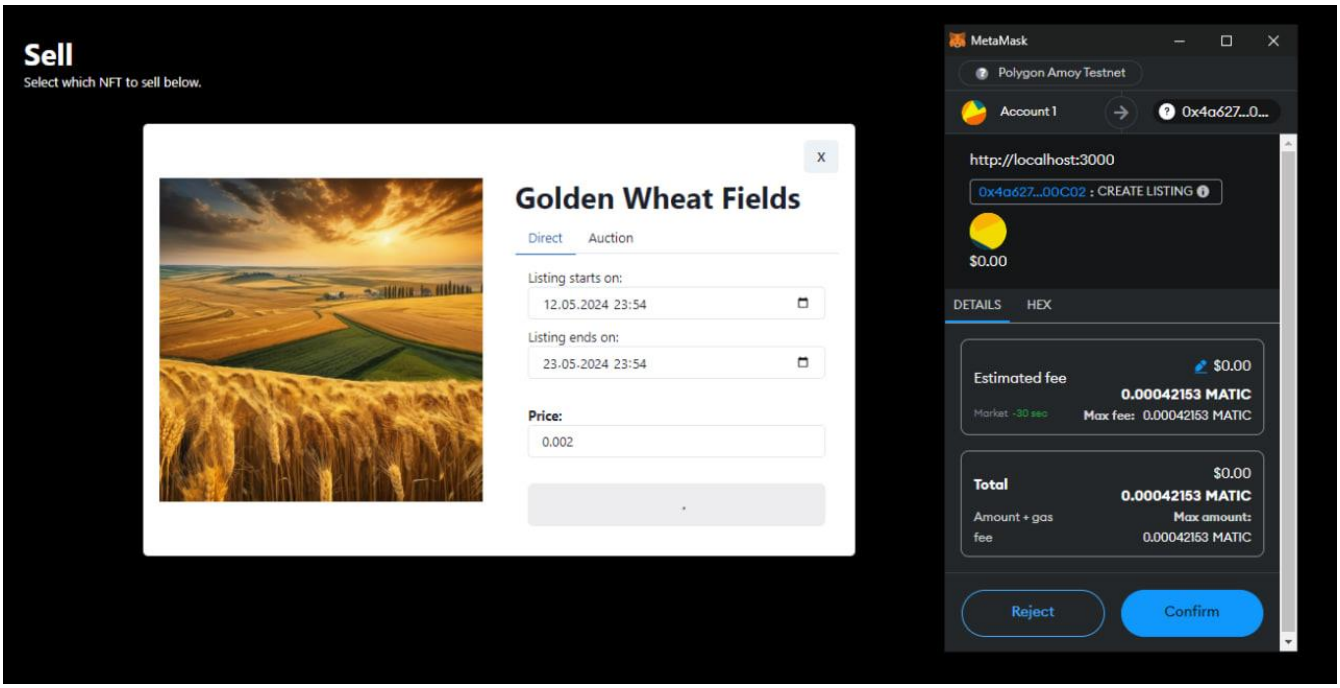

Рисунок 24 (Виставлення NFT на продаж)

Під час огляду токену, є можливість ознайомитися з метаданими активу, дізнатися хто його поточний власник та яка ціна купівлі. Після підтвердження та здійснення транзакції купівлі, придбані NFT відображаються користувачу в його особистому кабінеті.

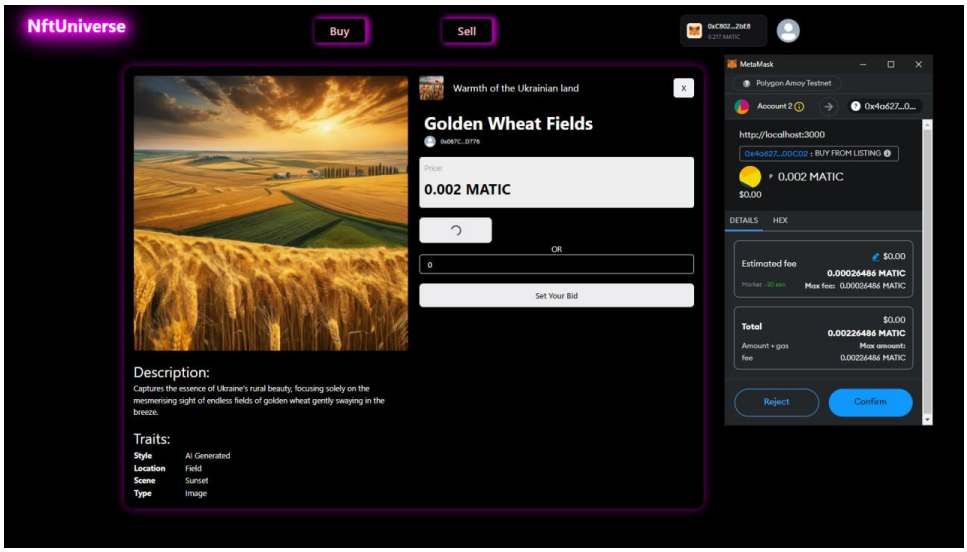

Рисунок 25 (Купівля NFT, підтвердження транзакції)

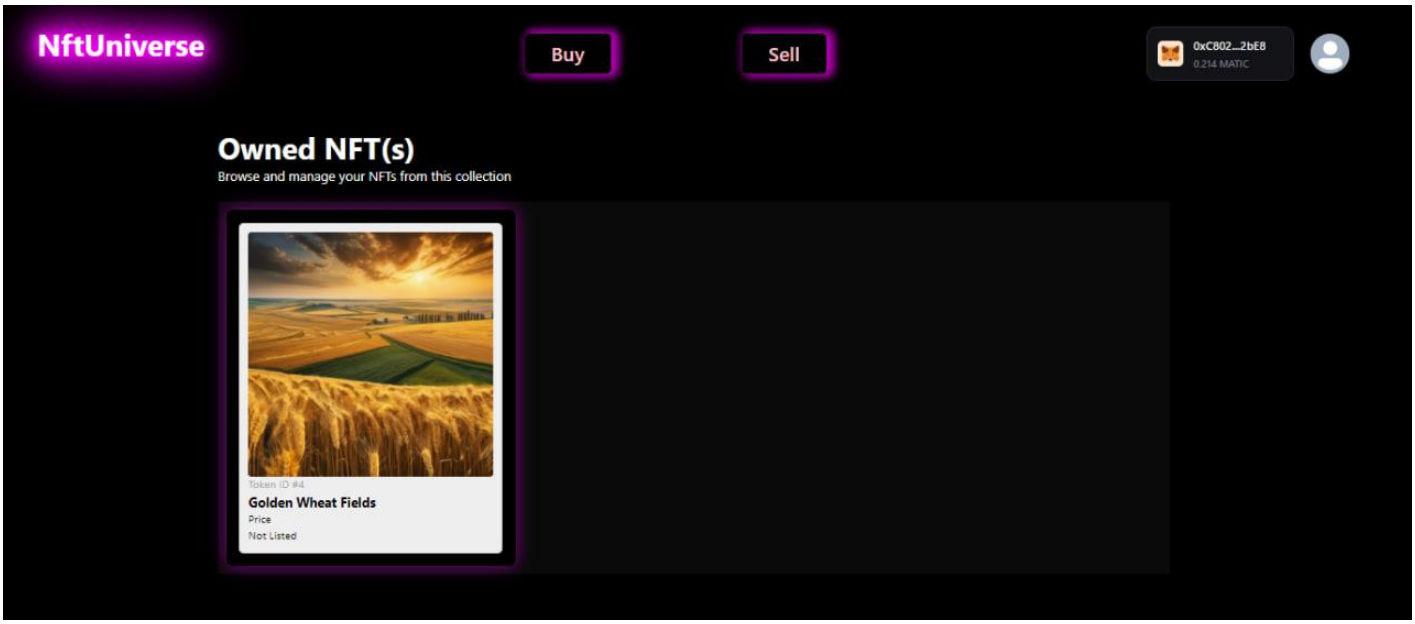

Рисунок 26 (Придбані NFT)

Створення аукціону відбувається за схожою схемою, заповнюються необхідні поля для часового проміжку проведення аукціону, розмір мінімальної ставки та ціна викупу токену, після чого з’являється спливаюче вікно для підтвердження транзакції через гаманець.

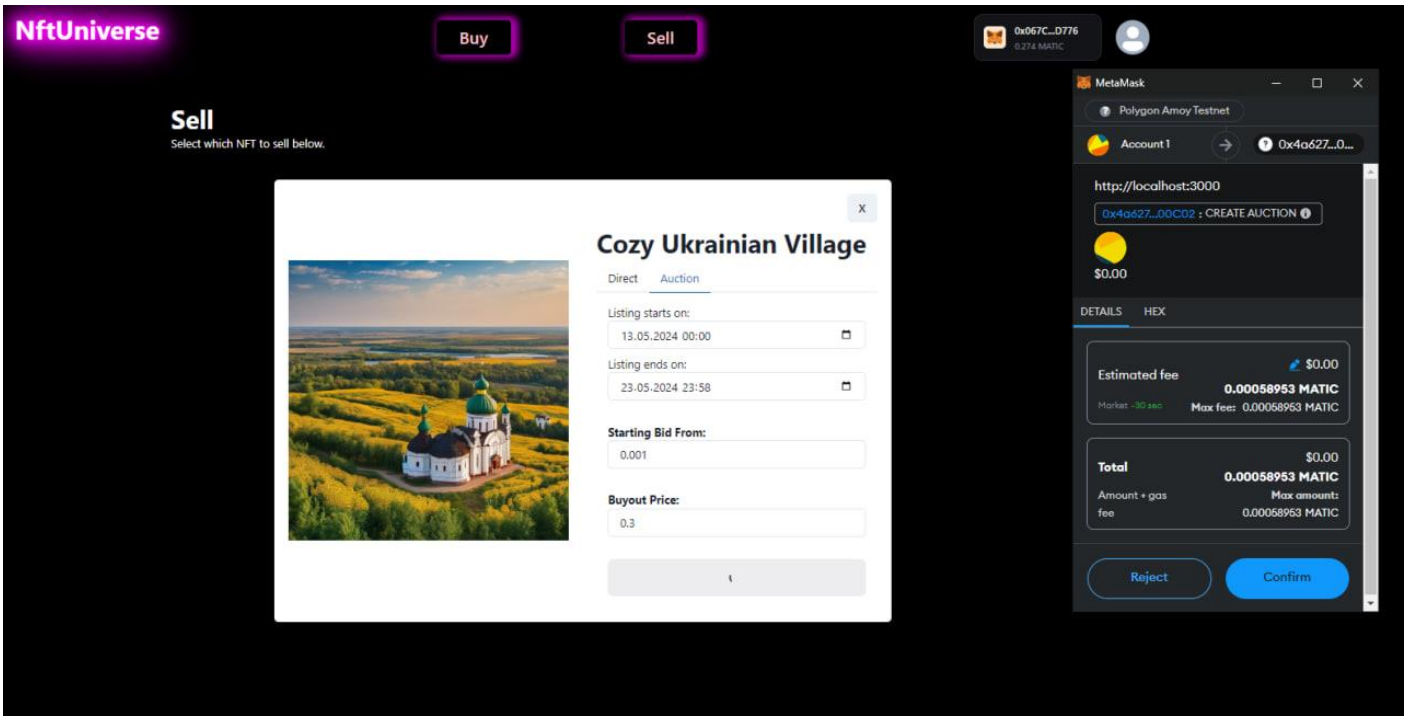

Рисунок 27 (Виставлення NFT на аукціон)

Коли NFT розміщено на аукціоні користувачі можуть встановлювати свою ставку на нього, після підтвердження транзакції сума мінімальнох ставки буде автоатично оновлена, як продемонстровано на рисунках 28 та 29. Це створено для того аби щоб не дезорієнтовувати інших учасників системи стосовно поточної ціни на токен, що бере участь в аукціоні.

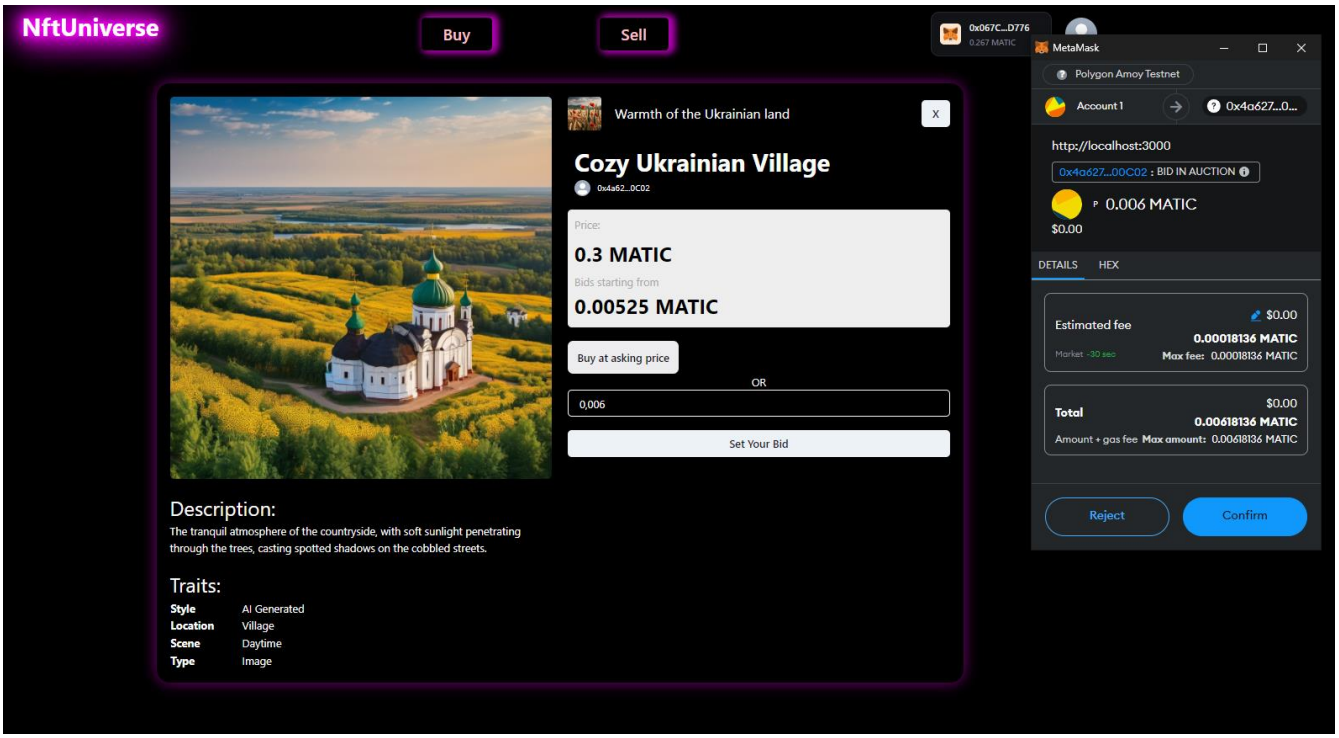

Рисунок 28 (Демонстрація встановлення власної ставки на аукціоні)

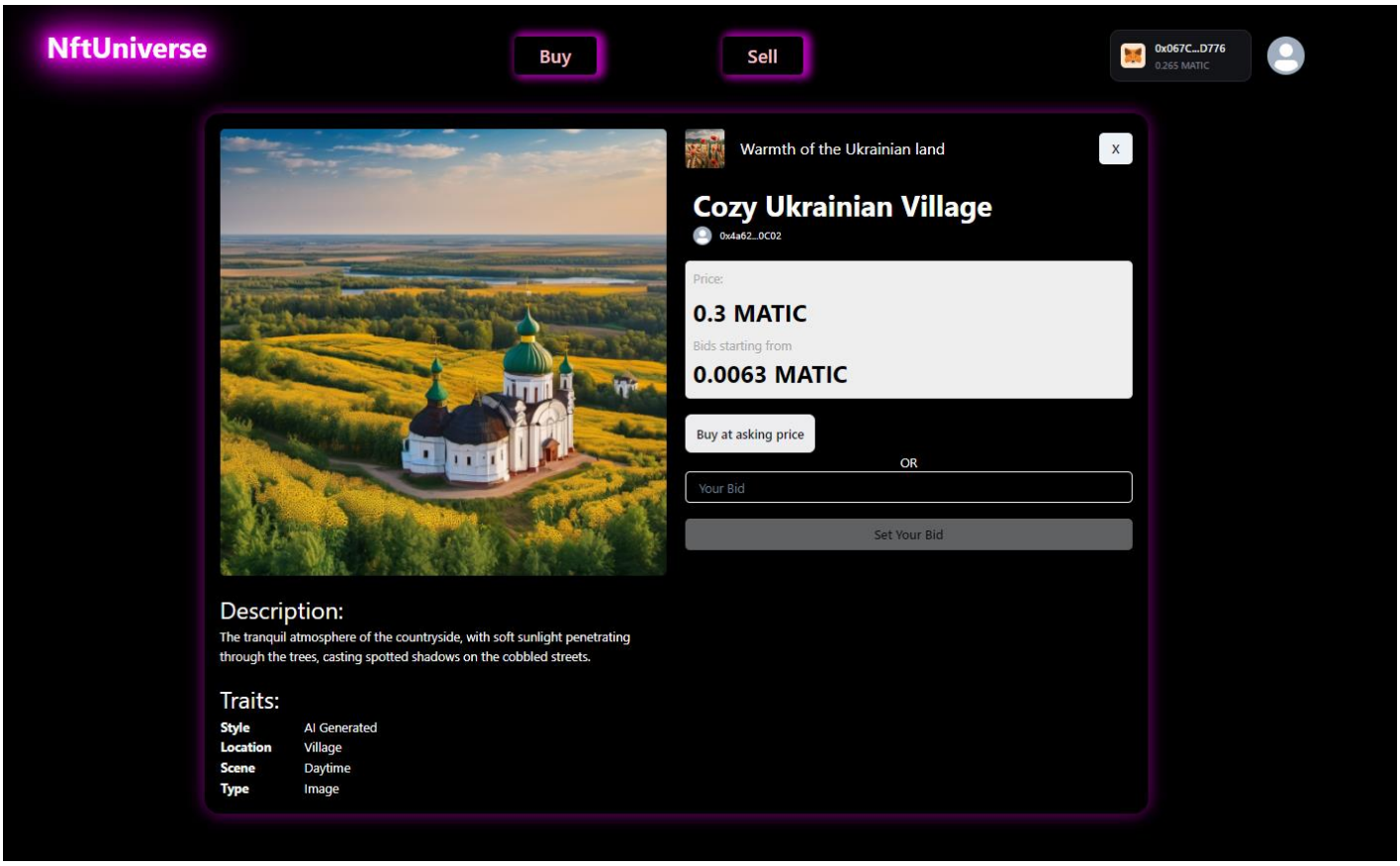

Рисунок 29 (Оновлення мінімальної ставки)

3.6 Шляхи розвитку та покращення NFT маркетплейсу

На даному етапі авторизація відбувається за допомогою гаманця Metamask, але для того аби підвищити рівень захищеності застосунку від шахраїв можна імплементувати створення акаунтів. Для входу в свій обліковий запис користувач може використовувати логін та пароль, але перш ніж здійснювати покупки або робити продажі, користувачу буде необхідно підтвердити свою особистість через гаманець Metamask. Такий функціонал також дозволить відслідковувати активність користувачів на платформі : як часто здійснюються продажі, які токени є більш популярними, як довго триває аукціон і т.д..

Таким чином маркетплейс буде забезпечуватися фінансовими ресурсами для зростання, вдосконалення і розвитку, а також для заробітку на своєму функціонуванні.

Можна імплементувати функціонал пошуку та фільтрації за конкретними метаданими, категоріями або назвою, щоб користувачі могли легко знаходити потрібні їм токени.

Висновки

Підводячи підсумки курсової роботи можна сформувати наступні висновки.
При роботі над практичною частиною було створено власну колекцію NFT, зображення для якої було згенеровано за допомогою штучного інтелекту. Для збереження малюнків було використано Filebase, що інтегровано з протоколом IPFS.
Було розроблено маркетплейс для розміщення створеної NFT колекції, що дозволяє користувачам купувати або продавати токени та брати участь в аукціонах. Авторизацію в застосунку було здійснено за допомогою гаманця Metamask, що є одним з найбільш надійних інструментів, який забезпечує високий рівень безпеки та довіру користувачів до безпечності їхніх транзакцій та особистих даних. При розробці застосунку використано платформу Thirdweb для деплою смарт-контрактів NFT колекції та маркетплейсу, а також інструменти для взаємодії з цими смарт-контрактами. Інструменти React SDK, які надає ця платформа, було використано для створення яскравого та функціонального інтерфейсу застосунку.
Розроблений маркетплейс визначив кілька напрямків для подальшого розвитку та удосконалення системи, включаючи покращення функціоналу за допомогою впровадження пошуку, встановлення реєстрації в системі та отримання заробітку через комісію.

Список використаних джерел

1. https://www.researchgate.net/publication/372495358_Survey_on_blockchain-based_non-fungible_tokens_History_technologies_standards_and_open_challenges
2. https://www.irjmets.com/uploadedfiles/paper/issue_9_september_2022/30345/final/fin_irjmets1664649599.pdf
3. https://www.investopedia.com/web-20-web-30-5208698
4. https://ieeexplore.ieee.org/document/8726782
5. https://www.researchgate.net/publication/378807496_Blockchain_Technology
6. https://blog.cfte.education/what-is-cryptography-in-blockchain/
7. https://www.gate.io/learn/articles/what-is-hashing-in-blockchain/864
8. https://www.researchgate.net/publication/369672172_Cryptography_of_Blockchain
9. https://www.geeksforgeeks.org/blockchain-merkle-trees/
10. https://coinmarketcap.com/academy/glossary/merkle-tree
11. https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp
12. https://coinmarketcap.com/academy/glossary/proof-of-authority-poa
13. https://www.researchgate.net/publication/350846372_A_Blockchain-based_Approach_to_Secure_Cloud_Connected_IoT_Devices
14. https://hal.science/hal-04401093/file/Non_Fungible_Tokens_A_review.pdf
15. https://arxiv.org/pdf/2209.14395
16. https://research.aimultiple.com/smart-contract-nft/

17. https://code.visualstudio.com/docs/editor/whyvscode

18.https://thirdweb.com/mission

19.https://metamask.io/

20.https://www.oklink.com/amoy

21.https://portal.thirdweb.com/react/v4

22.https://portal.thirdweb.com/react/v4/components

23. https://cds.cern.ch/record/369245/files/dd-89-001.pdf

24. https://portal.thirdweb.com/

Додаток А. Код застосунку

Navbar.tsx

```
import { Avatar, Box, Button, Flex, Heading, Link, Text } from "@chakra-ui/react";
import { ConnectWallet, useAddress } from "@thirdweb-dev/react";
import NextLink from 'next/link';
```

```
import React from "react";

export function Navbar(){

  const address = useAddress();

  return (
    <Box bg="black" py={"40px"} px={"40px"}>
      <Flex maxW={"1600px"} m={"auto"} justifyContent={"space-between"} color="white">
        <NextLink href='/' passHref>
          <Link>
            <Heading
              color="white"
              textShadow="0px 0px 8px #ff00ff, 0px 0px 16px #ff00ff, 0px 0px 24px #ff00ff,
0px 0px 32px #ff00ff, 0px 0px 40px #ff00ff, 0px 0px 48px #ff00ff, 0px 0px 56px #ff00ff, 0px
0px 64px #ff00ff"
            >
              NftUniverse
            </Heading>
          </Link>
        </NextLink>
        <Flex direction={"row"} alignItems="center">
          <Link as={NextLink} href='/buy' mx={20}>
            <Button
              fontSize="2xl"
              px={8}
              py={6}
              color="pink"
              bg="black"
              textDecoration="none"
              _hover={{
                bg: "gray.800",
                textDecoration: "none",
              }}
              boxShadow="5px 0px 10px 5px rgba(255, 0, 255, 0.7)"
            >
              Buy
            </Button>
          </Link>
          <Link as={NextLink} href='/sell' mx={20}>
            <Button
              fontSize="2xl"
```

```
              px={8}
              py={6}
              color="pink"
              bg="black"
              textDecoration="none"
              _hover={{
                bg: "gray.800",
                textDecoration: "none",
              }}
              boxShadow="5px 0px 10px 5px rgba(255, 0, 255, 0.7)"
            >
              Sell
            </Button>
          </Link>
        </Flex>
        <Flex dir={"row"} alignItems={"center"}>
          <ConnectWallet btnTitle="Login"/>
          {address && (
            <NextLink href={`/profile/${address}`} passHref>
              <Link ml={"20px"}>
                <Avatar src='https://bit.ly/broken-link'/>
              </Link>
            </NextLink>
          )}
        </Flex>
      </Flex>
    </Box>
  )
}
```

NFT.tsx

```
import React from "react";
import { NFT } from "@thirdweb-dev/sdk";
import {
  MARKETPLACE_ADDRESS,
  NFT_COLLECTION_ADDRESS
} from "../const/addresses";
```

```
import { ThirdwebNftMedia, useContract, useValidDirectListings, useValidEnglishAuctions }
from "@thirdweb-dev/react";
import { Box, Flex, Skeleton, Text } from "@chakra-ui/react";

type Props = {
  nft: NFT;
};

export default function NFTComponent({ nft }: Props) {
  const { contract: marketplace, isLoading: loadingMarketplace } =
useContract(MARKETPLACE_ADDRESS, "marketplace-v3");

  const { data: directListing, isLoading: loadingDirectListing } =
    useValidDirectListings(marketplace, {
      tokenContract: NFT_COLLECTION_ADDRESS,
      tokenId: nft.metadata.id,
    });

  const { data: auctionListing, isLoading: loadingAuction} =
    useValidEnglishAuctions(marketplace, {
      tokenContract: NFT_COLLECTION_ADDRESS,
      tokenId: nft.metadata.id,
    });

  return (
    <Box boxShadow="0px 0px 20px rgba(255, 0, 255, 0.5)" p="4" borderRadius="lg"
bg="black">
      <Flex direction="column" backgroundColor="#EEE" justifyContent="center" padding="2.5"
borderRadius="6px" borderColor="lightgray" borderWidth={1}>
        <Box borderRadius="4px" overflow="hidden">
          <ThirdwebNftMedia metadata={nft.metadata} height="100%" width="100%" />
        </Box>
        <Text fontSize="small" color="darkgray">Token ID #{nft.metadata.id}</Text>
        <Text fontWeight="bold" color="black">{nft.metadata.name}</Text>

        <Box>
          {loadingMarketplace || loadingDirectListing || loadingAuction ? (
            <Skeleton></Skeleton>
          ) : directListing && directListing[0] ? (
            <Box>
              <Flex direction="column">
```

```
                <Text fontSize="small" color="black">Price</Text>
                <Text fontSize="small"
color="black">{`${directListing[0]?.currencyValuePerToken.displayValue}
${directListing[0]?.currencyValuePerToken.symbol}`}</Text>
              </Flex>
            </Box>
          ) : auctionListing && auctionListing[0] ? (
            <Box>
              <Flex direction="column">
                <Text fontSize="small" color="black">Minimum Bid</Text>
                <Text fontSize="small"
color="black">{`${auctionListing[0]?.minimumBidCurrencyValue.displayValue}
${auctionListing[0]?.minimumBidCurrencyValue.symbol}`}</Text>
              </Flex>
            </Box>
          ) : (
            <Box>
              <Flex direction="column">
                <Text fontSize="small" color="black">Price</Text>
                <Text fontSize="small" color="black">Not Listed</Text>
              </Flex>
            </Box>
          )}
        </Box>
      </Flex>
    </Box>
  );
}
```

NFTGrid.tsx

```
import type {NFT as NFTType } from "@thirdweb-dev/sdk";
import { SimpleGrid, Skeleton, Text } from "@chakra-ui/react";
import React from "react";
import NFT from "./NFT";
import Link from "next/link";
import { NFT_COLLECTION_ADDRESS } from "../const/addresses";

type Properties = {
```

```
  isLoading: boolean;
  data: NFTType[] | undefined;
  overrideOnclickBehavior?: (nft: NFTType) => void;
  emptyText?: string;
  textColor?: string;
};

export default function NFTGrid({
                                  isLoading,
                                  data,
                                  overrideOnclickBehavior,
                                  emptyText = "No NFTs found",
                                  textColor = "white",
                                }: Properties) {
  return (
    <SimpleGrid columns={3} spacing={10} w={"100%"} padding={2.5} my={5} bg="#0a0a0a"
color={textColor} > {/* Використовуйте textColor */}
      {isLoading ? (
        [...Array(20)].map((_, index) => (
          <Skeleton key={index} height={"312px"} width={"100%"} />
        ))
      ) : data && data.length > 0 ? (
        data.map((nft) =>
          !overrideOnclickBehavior ? (
            <Link
              href={`/token/${NFT_COLLECTION_ADDRESS}/${nft.metadata.id}`}
              key={nft.metadata.id}
            >
              <NFT nft={nft} />
            </Link>
          ) : (
            <div
              key={nft.metadata.id}
              onClick={() => overrideOnclickBehavior(nft)}
            >
              <NFT nft={nft} />
            </div>
          ))
      ) : (
        <Text>{emptyText}</Text>
      )}
    </SimpleGrid>
```

```
  );
}
```

Saleinfo.tsx

```
import React from "react";
import { NFT as NFTType } from "@thirdweb-dev/sdk";
import { useRouter } from "next/router";
import { useForm } from "react-hook-form";
import { Web3Button, useContract, useCreateAuctionListing, useCreateDirectListing } from
"@thirdweb-dev/react";
import { MARKETPLACE_ADDRESS, NFT_COLLECTION_ADDRESS } from "../const/addresses";
import { Box, Input, Stack, Tabs, TabList, Tab, TabPanels, TabPanel, Text } from "@chakra-
ui/react";

type Properties = {
  nft: NFTType
};

type DirectFormData = {
  nftContractAddress: string;
  tokenId: string;
  price: string;
  startDate: Date;
  endDate: Date;
};

type AuctionFormData = {
  nftContractAddress: string;
  tokenId: string;
  floorPrice: string;
  buyOutPrice: string;
  startDate: Date;
  endDate: Date;
};

export default function SaleInfo({ nft }: Properties) {
  const router = useRouter();

  const { contract: marketplace } = useContract(MARKETPLACE_ADDRESS, "marketplace-v3");
```

```
  const { contract: nftCollection } = useContract(NFT_COLLECTION_ADDRESS);
  const { mutateAsync: createDirectListing } = useCreateDirectListing(marketplace);
  const { mutateAsync: createAuctionListing } = useCreateAuctionListing(marketplace);

  async function checkAndProvideApproval() {
    const hasApproval = await nftCollection?.call(
      "isApprovedForAll",
      [nft.owner, MARKETPLACE_ADDRESS]
    );

    if (!hasApproval) {
      const transactionResult = await nftCollection?.call(
        "setApprovalForAll",
        [MARKETPLACE_ADDRESS, true]
      );

      if (transactionResult) {
        console.log("Approval provided");
      }
    }

    return true;
  }

  const { register: registerDirect, handleSubmit: handleSubmitDirect } =
useForm<DirectFormData>({
    defaultValues: {
      nftContractAddress: NFT_COLLECTION_ADDRESS,
      tokenId: nft.metadata.id,
      price: "0",
      startDate: new Date(),
      endDate: new Date(),
    },
  });

  async function handleSubmissionDirect(data: DirectFormData) {
    await checkAndProvideApproval();
    const transactionResult = await createDirectListing({
      assetContractAddress: data.nftContractAddress,
      tokenId: data.tokenId,
      pricePerToken: data.price,
```

```
      startTimestamp: new Date(data.startDate),
      endTimestamp: new Date(data.endDate),
    });

    return transactionResult;
  }

  const { register: registerAuction, handleSubmit: handleSubmitAuction } =
useForm<AuctionFormData>({
    defaultValues: {
      nftContractAddress: NFT_COLLECTION_ADDRESS,
      tokenId: nft.metadata.id,
      startDate: new Date(),
      endDate: new Date(),
      floorPrice: "0",
      buyOutPrice: "0",
    },
  });

  async function handleSubmissionAuction(data: AuctionFormData) {
    await checkAndProvideApproval();
    const transactionResult = await createAuctionListing({
      assetContractAddress: data.nftContractAddress,
      tokenId: data.tokenId,
      minimumBidAmount: data.floorPrice,
      buyoutBidAmount: data.buyOutPrice,
      startTimestamp: new Date(data.startDate),
      endTimestamp: new Date(data.endDate),
    });

    return transactionResult;
  }

  return (
    <Tabs>
      <TabList>
        <Tab>Direct</Tab>
        <Tab>Auction</Tab>
      </TabList>
      <TabPanels>
        <TabPanel>
```

```
          <Stack spacing={8}>
            <Box>
              <Text>Listing starts on:</Text>
              <Input
                placeholder="Select Date and Time"
                size="md"
                type="datetime-local"
                {...registerDirect("startDate")}
              />
              <Text mt={2}>Listing ends on:</Text>
              <Input
                placeholder="Select Date and Time"
                size="md"
                type="datetime-local"
                {...registerDirect("endDate")}
              />
            </Box>
            <Box>
              <Text fontWeight="bold">Price:</Text>
              <Input
                placeholder="0"
                size="md"
                type="number"
                {...registerDirect("price")}
              />
            </Box>
            <Web3Button
              contractAddress={MARKETPLACE_ADDRESS}
              action={async () => {
                await handleSubmitDirect(handleSubmissionDirect)();
              }}
              onSuccess={(transactionResult) => {
                router.push(`/token/${NFT_COLLECTION_ADDRESS}/${nft.metadata.id}`);
              }}
            >
              Create Direct Listing
            </Web3Button>
          </Stack>
        </TabPanel>
        <TabPanel>
          <Stack spacing={8}>
            <Box>
```

```
            <Text>Listing starts on:</Text>
            <Input
              placeholder="Select Date and Time"
              size="md"
              type="datetime-local"
              {...registerAuction("startDate")}
            />
            <Text mt={2}>Listing ends on:</Text>
            <Input
              placeholder="Select Date and Time"
              size="md"
              type="datetime-local"
              {...registerAuction("endDate")}
            />
          </Box>
          <Box>
            <Text fontWeight="bold">Starting Bid From:</Text>
            <Input
              placeholder="0"
              size="md"
              type="number"
              {...registerAuction("floorPrice")}
            />
          </Box>
          <Box>
            <Text fontWeight="bold">Buyout Price:</Text>
            <Input
              placeholder="0"
              size="md"
              type="number"
              {...registerAuction("buyOutPrice")}
            />
          </Box>
          <Web3Button

            contractAddress={MARKETPLACE_ADDRESS}
            action={async () => {
              await handleSubmitAuction(handleSubmissionAuction)();
            }}
            onSuccess={(transactionResult) => {
              router.push(`/token/${NFT_COLLECTION_ADDRESS}/${nft.metadata.id}`);
            }}
```

```
              style={{ color: "red" }}

            >
              Create Auction Listing
            </Web3Button>
          </Stack>
        </TabPanel>
      </TabPanels>
    </Tabs>
  );
}
```

[addreses].ts

```
import {  PolygonAmoyTestnet } from "@thirdweb-dev/chains";
export const NETWORK = PolygonAmoyTestnet;
export const MARKETPLACE_ADDRESS = "0x4a62787e8BffA8a5ADCfC38514fBC6fBecF00C02"

export const NFT_COLLECTION_ADDRESS = "0xb2B53a34359E8b81891E7286aDeD4B162722C0cB"
```

[address].ts

```
import { Container, Heading, Text, Box } from "@chakra-ui/react";
import { useContract, useOwnedNFTs } from "@thirdweb-dev/react";
import React from "react";
import { NFT_COLLECTION_ADDRESS } from "../../const/addresses";
import { useRouter } from "next/router";
import NFTGrid from "../../components/NFTGrid";
import { use } from "h3";

export default function PersonalProfile(){
  const router = useRouter();
  const { contract: nftCollection } = useContract(NFT_COLLECTION_ADDRESS);

  const {data: ownedNfts, isLoading: loadingOwnedNfts} = useOwnedNFTs(
    nftCollection,
    router.query.address as string
  );
```

```
  return(
    <Box bg="black" minH="120vh" color="white" mt={0}>

    <Container maxW={"1200px"} p={5} bg={"black"} color={"white"}>
      <Heading>{"Owned NFT(s)"}</Heading>
      <Text>Browse and manage your NFTs from this collection</Text>
      <NFTGrid
        data={ownedNfts}
        isLoading={loadingOwnedNfts}
        emptyText={"You don't own any NFTs yet from this collection."}
      />

    </Container>
    </Box>

  )
}
```

[tokenId].tsx

```
import React, { useState } from "react";
import { Box, Container, SimpleGrid, Stack, Text, Flex, Avatar, Skeleton, Input, Button }
from "@chakra-ui/react";
import {
  MediaRenderer,
  ThirdwebNftMedia,
  useContract,
  useMinimumNextBid,
  useValidDirectListings,
  useValidEnglishAuctions,
  Web3Button
} from "@thirdweb-dev/react";
import { NFT, ThirdwebSDK } from "@thirdweb-dev/sdk";
import { MARKETPLACE_ADDRESS, NFT_COLLECTION_ADDRESS, NETWORK } from
"../../../const/addresses";
import { GetStaticPaths, GetStaticProps } from "next";
import Link from "next/link";
import { Spacer } from "@chakra-ui/react";
```

```
type Properties = {
  nft: NFT;
  contractMetadata: any;
};

export default function TokenPage({ nft, contractMetadata }: Properties) {
  const { contract: marketplace, isLoading: loadingMarketplace } =
useContract(MARKETPLACE_ADDRESS, "marketplace-v3");

  const [bidPrice, setBidPrice] = useState<number>(0);

  const { data: directListing, isLoading: loadingDirectListing } =
useValidDirectListings(marketplace, {
    tokenContract: NFT_COLLECTION_ADDRESS,
    tokenId: nft.metadata.id,
  });

  // Отримання аукціонного списку
  const { data: auctionListing, isLoading: loadingAuction } =
useValidEnglishAuctions(marketplace, {
    tokenContract: NFT_COLLECTION_ADDRESS,
    tokenId: nft.metadata.id,
  });

  // Використання хука useMinimumNextBid для отримання мінімальної наступної ставки
  const { data: minimumNextBid, isLoading: loadingMinimumNextBid } = useMinimumNextBid(
    marketplace,
    auctionListing && auctionListing[0] ? auctionListing[0].id : null
  );

  async function buyToken() {
    let transactionRes;

    if (directListing && directListing[0]) {
      transactionRes = await
marketplace?.directListings.buyFromListing(directListing[0].id, 1);
    } else {
      throw new Error("No direct listing found");
    }

    return transactionRes;
```

```
  }

  async function submitBidOffer() {
    if (!bidPrice || bidPrice <= 0) {
      return;
    }

    if (auctionListing && auctionListing[0]) {
      await marketplace?.englishAuctions.makeBid(auctionListing[0].id, bidPrice);
      // Optionally, you can update the bid price state or fetch the latest bid
    } else {
      throw new Error("No auction listing found");
    }
  }

  return (
    <Box bg="black" minH="120vh" color="white" mt={0}>
      <Container borderRadius="20px" boxShadow="0px 0px 20px rgba(255, 0, 255, 0.5)"
maxW={"1200px"} mb={20} p={5} bg={"black"}>
        <SimpleGrid columns={2} spacing={6}>
          <Stack spacing={"20px"}>
            <Box borderRadius={"6px"} overflow={"hidden"}>
              <Skeleton isLoaded={!loadingMarketplace && !loadingDirectListing}>
                <ThirdwebNftMedia metadata={nft.metadata} width="100%" height="100%" />
              </Skeleton>
            </Box>
            <Box>
              <Text fontSize={"3xl"}>Description:</Text>
              <Text>{nft.metadata.description}</Text>
            </Box>
            <Box>
              <Text fontSize={"3xl"}>Traits:</Text>
              <table>
                <tbody>
                {Object.entries(nft?.metadata?.attributes || {}).map(([key, value]:
[string, any], index) => (
                  <tr key={key} style={{ marginBottom: "50px" }}>
                    <td>
                      <Text fontSize={"l"} fontWeight={"bold"}>{value.trait_type}</Text>
                    </td>
                    <td>
                      <Text fontSize={"l"} ml={10}>{value.value}</Text>
```

```
                  </td>
                </tr>
              ))}
              </tbody>
            </table>
          </Box>
        </Stack>
        <Stack spacing={"20px"}>
          {contractMetadata && (
            <Flex alignItems={"center"}>
              <Box borderRadius={"4px"} overflow={"hidden"} mr={"20px"}>
                <MediaRenderer src={contractMetadata.image} height="50px" width="50px" />
              </Box>
              <Text fontSize={"xl"}>{contractMetadata.name}</Text>
              <Spacer />
              <Link href='/buy'>
                <Button>X</Button>
              </Link>
            </Flex>
          )}
          <Box mx={2.5}>
            <Text fontSize={"4xl"} fontWeight={"bold"}>{nft.metadata.name}</Text>
            <Link href={`/profile/${nft.owner}`}>
              <Flex direction={"row"} alignItems={"center"}>
                <Avatar src='https://bit.ly/broken-link' h={"24px"} w={"24px"}
mr={"10px"} />
                <Text fontSize={"small"}> {nft.owner.slice(0, 6)}...{nft.owner.slice(-
4)}</Text>
              </Flex>
            </Link>
          </Box>
          <Stack backgroundColor={"#EEE"} p={2.5} borderRadius={"6px"}>
            <Text color={"darkgray"}>Price:</Text>
            <Skeleton isLoaded={!loadingMarketplace && !loadingDirectListing}>
              {directListing && directListing[0] ? (
                <Text fontSize={"3xl"} fontWeight={"bold"} >
                  <Text as="span"
color="black">{directListing[0]?.currencyValuePerToken.displayValue}</Text>
                  {" "}
                  <Text as="span"
color="black">{directListing[0]?.currencyValuePerToken.symbol}</Text>
                </Text>
```

```
              ) : auctionListing && auctionListing[0] ? (
                <Text fontSize={"3xl"} fontWeight={"bold"}>
                  <Text as="span"
color="black">{auctionListing[0]?.buyoutCurrencyValue.displayValue}</Text>
                  {" "}
                  <Text as="span"
color="black">{auctionListing[0]?.buyoutCurrencyValue.symbol}</Text>
                </Text>
              ) : (
                <Text fontSize={"3xl"} fontWeight={"bold"} color="black">Not for
sale</Text>
              )}
            </Skeleton>
            <Skeleton isLoaded={!loadingAuction}>
              {auctionListing && auctionListing[0] && (
                <Flex direction={"column"}>
                  <Text color={"darkgray"}>Bids starting from</Text>
                  <Text fontSize={"3xl"} fontWeight={"bold"} color={"black"}>
                    {minimumNextBid?.displayValue}
                    {" "}
                    {minimumNextBid?.symbol}
                  </Text>
                  <Text></Text>
                </Flex>
              )}
            </Skeleton>
          </Stack>

          <Skeleton isLoaded={!loadingMarketplace || !loadingDirectListing ||
!auctionListing}>
            <Web3Button
              contractAddress={MARKETPLACE_ADDRESS}
              action={async () => buyToken()}
              isDisabled={(!auctionListing || !auctionListing[0]) && (!directListing ||
!directListing[0])}
            >Buy at asking price</Web3Button>
            <Text align={"center"}>OR</Text>
            {auctionListing && auctionListing[0] && (
              <Flex direction={"column"}>
                <Input
                  mb={5}
                  type="number"
```

```
                    placeholder="Your Bid"
                    value={bidPrice}
                    color ={"white"}
                    border="1px solid white"
                    onChange={(e) => setBidPrice(parseFloat(e.target.value))}

                  />
                  <Button onClick={submitBidOffer} isDisabled={!bidPrice || bidPrice <=
0}>Set Your Bid</Button>
                </Flex>
              )}
            </Skeleton>
          </Stack>
        </SimpleGrid>
      </Container>
    </Box>
  )
}

export const getStaticProps: GetStaticProps = async (context) => {
  const tokenId = context.params?.tokenId as string;

  const sdk = new ThirdwebSDK(NETWORK);

  const contract = await sdk.getContract(NFT_COLLECTION_ADDRESS);

  const nft = await contract.erc721.get(tokenId);

  let contractMetadata;

  try {
    contractMetadata = await contract.metadata.get();
  } catch (e) {}

  return {
    props: {
      nft,
      contractMetadata: contractMetadata || null,
    },
    revalidate: 1,
  };
```

```
};

export const getStaticPaths: GetStaticPaths = async () => {
  const sdk = new ThirdwebSDK(NETWORK);

  const contract = await sdk.getContract(NFT_COLLECTION_ADDRESS);

  const nfts = await contract.erc721.getAll();

  const paths = nfts.map((nft) => {
    return {
      params: {
        contractAddress: NFT_COLLECTION_ADDRESS,
        tokenId: nft.metadata.id,
      },
    };
  });
  return {
    paths,
    fallback: "blocking",
  };
};
```

_app.tsx

```
import type { AppProps } from "next/app";
import { ThirdwebProvider } from "@thirdweb-dev/react";
import "../styles/globals.css";
import { ChakraProvider } from "@chakra-ui/react";
import { Navbar } from "../components/Navbar";
import { NETWORK } from "../const/addresses";

function App({ Component, pageProps }: AppProps) {
  return (
    <ThirdwebProvider activeChain={NETWORK}
clientId={process.env.NEXT_PUBLIC_TEMPLATE_CLIENT_ID} >
      <ChakraProvider>
        <Navbar/>
        <Component {...pageProps} />
      </ChakraProvider>
    </ThirdwebProvider>
  );
```

```
}

export default App;
```

buy.tsx

```
import React from "react";
import { Container, Heading, Text, Box } from "@chakra-ui/react";
import NFTGrid from "../components/NFTGrid";
import { NFT_COLLECTION_ADDRESS } from "../const/addresses";
import { useContract, useNFTs } from "@thirdweb-dev/react";
import Link from "next/link";

export default function Buy() {
  const { contract } = useContract(NFT_COLLECTION_ADDRESS);
  const { data, isLoading } = useNFTs(contract);
  return (
    <Box bg="black" minH="100vh" color="white">
      <Container maxW={"1200px"} p={5} bg="black" color="white">
        <Heading>Buy NFTs</Heading>
        <Text>Browse and buy NFTs from this collection.</Text>
        <NFTGrid
          isLoading={isLoading}
          data={data}
          emptyText={"No NFTs found"}

        />
      </Container>
    </Box>
  )
}
```

index.tsx

```
import { NextPage } from "next";
```

```
import { Container, Heading, Flex, Stack, Button, Box, Text } from "@chakra-ui/react";
import NextLink from "next/link";

const Home: NextPage = () => {
  return (
    <Box bg="black" minH="100vh" color="white">
      <Container maxW={"1200px"}>
        <Flex h={"80vh"} alignItems={"center"} justifyContent={"center"}>
          <Stack spacing={4} align={"center"}>
            <Heading fontSize="7xl">Marketplace</Heading>
            <Text fontSize="xl">Here you can buy, sell or auction digital art in the form
of NFTs</Text>
            <Text>Wish you a pleasant experience</Text>

            <Button
              as={NextLink}
              href="/buy"
              boxShadow="0px 8px 12px rgba(255, 0, 255, 0.3)"
              _hover={{
                boxShadow: "10px 10px 20px rgba(255, 0, 255, 0.4)"
              }}
              colorScheme="pink"
              fontSize="2xl"
            >
              Shop NFTs
            </Button>
          </Stack>
        </Flex>
      </Container>
    </Box>
  );
};

export default Home;
```

sell.tss

```
import { Box, Button, Card, Container, Flex, Heading, SimpleGrid, Stack, Text } from
"@chakra-ui/react";
```

```
import { ThirdwebNftMedia, useAddress, useContract, useNFT, useOwnedNFTs } from "@thirdweb-
dev/react";
import React, { useState } from "react";
import { NFT_COLLECTION_ADDRESS } from "../const/addresses";
import type { NFT as NFTType } from "@thirdweb-dev/sdk";
import NFTGrid from "../components/NFTGrid";
import SaleInfo from "../components/SaleInfo";
import Link from "next/link";

export default function Sell() {
  const { contract } = useContract(NFT_COLLECTION_ADDRESS)
  const address = useAddress();
  const { data, isLoading } = useOwnedNFTs(contract, address);

  const [selectedNFT, setSelectedNFT] = useState<NFTType>();

  return (
    <Box bg="black" minH="100vh" color="white">
      <Container maxW={"1200px"} p={5} bg="black" color="white">
        <Heading>Sell</Heading>
      <Text>Select which NFT to sell below.</Text>

      {!selectedNFT ? (
        <NFTGrid
          data={data}
          isLoading={isLoading}
          overrideOnclickBehavior={(nft) => {
            setSelectedNFT(nft);
          }}
          emptyText={"You don't own any NFTs yet from this collection"}
        />
      ) : (
        <Flex justifyContent={"center"} my={10}>
          <Card w={"75%"}>
            <SimpleGrid columns={2} spacing={10} p={5}>
              <ThirdwebNftMedia
                metadata={selectedNFT.metadata}
                width="100%"
                height="100%"
              />
              <Stack>
```

```
                <Flex justifyContent={"right"}>
                  <Button
                    onClick={() => {
                      setSelectedNFT(undefined);
                    }}
                  >X</Button>
                </Flex>
                <Heading>{selectedNFT.metadata.name}</Heading>
                <SaleInfo
                  nft={selectedNFT}
                />
              </Stack>
            </SimpleGrid>
          </Card>
        </Flex>
      )}
    </Container>
    </Box>
  )
}
```

Додаток Б. Компіляція застосунку

```
PS C:\Users\Lutsiuk.Ivanna\UniDocs\nftuniverse\nftuniverse> yarn dev
yarn run v1.22.22
$ next dev
  ▲ Next.js 13.5.6
  - Local:        http://localhost:3000

 ✓ Ready in 1527ms
 ○ Compiling / ...
 ✓ Compiled / in 3s (3216 modules)
No API key. Please provide a clientId. It is required to access thirdweb's services. You can create a key at https://thirdweb.com/create-api-key
 ⚠ Fast Refresh had to perform a full reload. Read more: https://nextjs.org/docs/messages/fast-refresh-reload
No API key. Please provide a secretKey. It is required to access thirdweb's services. You can create a key at https://thirdweb.com/create-api-key
 ✓ Compiled /buy in 460ms (3224 modules)
 ✓ Compiled /sell in 307ms (3274 modules)
No API key. Please provide a secretKey. It is required to access thirdweb's services. You can create a key at https://thirdweb.com/create-api-key
 ✓ Compiled /profile/[address] in 357ms (3278 modules)
```