

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра
на тему: «Методи та технології підвищення рівня
захищеності криптоактивів»

Виконав студент 2-го курсу, групи ЗКБм-22-1
спеціальності 125 «Кібербезпека»

 М.О. Бацманов

Керівник, доцент кафедри комп'ютерних наук,
кандидат технічних наук, доцент

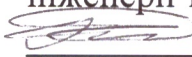
 І.І. Сугоняк

Рецензент, професор кафедри комп'ютерних наук,
доктор технічних наук, професор

 В.П. Квасніков

Житомир – 2023

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
 Андрій Єфіменко
13 10 2023 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу

Студент: Бацманов Максим Олександрович

Тема роботи: Методи та технології підвищення рівня захищеності
криптоактивів

Затверджена Наказом університету від «13» жовтня 2023 р. № 572/с

Термін здачі студентом закінченої роботи 11.12 2023 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є новітні методи та технології підвищення рівня захищеності криптоактивів. Предметом дослідження є технологія NFT яка допоможе забезпечити надійність крипто інвестицій.

Обладнання та програмне забезпечення:

1. Visual Studio Code
2. Solidity
3. NFT
4. Ganache
5. Truffle

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Сугоняк І.І.	13.10.2023	13.10.2023
2	Сугоняк І.І.	13.10.2023	13.10.2023
3	Сугоняк І.І.	13.10.2023	13.10.2023

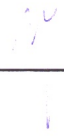
Студент  М.О. Бацманов

Керівник  Сугоняк І.І.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	01.03.2023	виконано
2	Розробка плану роботи	02.04.2023	виконано
3	Розробка розгорнутого плану	08.04.2023	виконано
4	Подача чернетки керівнику для перевірки	10.04.2023	виконано
5	Доопрацювання роботи та перевірка керівником	12.04.2023	виконано
6	Затвердження графічних матеріалів у керівника	19.04.2023	виконано
7	Проведення попереднього захисту	20.06.2023	виконано
8	Отримання відгуку керівника	20.06.2023	виконано
9	Допуск завідувача кафедри	20.06.2023	виконано
10	Подання роботи на рецензування	20.06.2023	виконано

Студент  М.О. Бацманов

Керівник  І.І. Сугоняк

АНОТАЦІЯ

Завдання кваліфікаційної роботи магістра полягає у вивченні та обґрунтуванні вибору інструментів, програмо-апаратних рішень, та методів захисту криптоактивів, побудувати зручне рішення та захищене рішення для інвестицій.

Кваліфікаційна робота магістра на тему «Методи та технології підвищення рівня захищеності криптоактивів » складається з інженерного, логічного, аналітичного, та теоретичного комплексів та пояснювальної записки.

Пояснювальна записка до випускної кваліфікаційної роботи містить в собі 102 сторінки, 1 таблицю та 36 рисунків. Список використаних джерел містить 19 найменування і займає 2 сторінки.

Ключові слова: Solidity, Smart Contract, ERC721, NFT, Ethereum, Web3, Ganache, Blockchain.

ANNOTATION

The task of the master's thesis is to study and justify the choice of tools, software and hardware solutions, and methods for protecting cryptoassets, to build a convenient solution and a secure solution for investment.

The master's qualification work on the topic "Methods and technologies for increasing the level of security of crypto assets" consists of engineering, logical, analytical, and theoretical complexes and an explanatory note.

The explanatory note to the qualification work contains 102 pages, 1 table and 36 figures. The list of used sources contains 19 names and occupies 2 pages.

Keywords: Solidity, Smart Contract, ERC721, NFT, Ethereum, Web3, Ganache, Blockchain.

Зміст

ВСТУП	8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	9
1.1 Актуальність технології blockchain	9
1.2 Аналіз технології Ethereum	10
1.3 Розбір поняття смарт-контракт	11
1.4 Актуальність NFT в сучасному світі	12
1.5 Аналіз ринку NFT маркетплейсів	15
1.6 Постановка завдання на розробку	17
РОЗДІЛ 2 ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ. РЕАЛІЗАЦІЯ ПРОЄКТУ	19
2.1 Вибір технологій	19
2.1.1 Next.js	19
2.1.2 Metamask	22
2.1.3 Solidity	27
2.1.4 Truffle	29
2.1.5 Ganache	31
2.2 Розробка смарт-контракту	33
2.3 Компіляція та деплой смарт контракту	41
2.4 Розробка веб інтерфейсу для взаємодії з контрактом	43
2.4.1 Створення дизайну	43
2.4.2 Створення веб додатку	45
РОЗДІЛ 3 ТЕСТУВАННЯ ПРОЄКТУ, МІНТ NFT, ВІДСТЕЖЕННЯ ПОДІЙ ТА ТРАНЗАКЦІЙ	50
3.1 Тестування проєкту	50
3.2 Відстеження подій блокчейну	52
ВИСНОВКИ	54
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	55
ДОДАТКИ	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ETH - Ether криптовалюта

EVM - Ethereum Virtual Machine

ВСТУП

Актуальність теми. Протягом останніх років блокчейн значно розширив свої можливості і вплив на реальний світ. Зростає інтерес до криптовалют, завдяки системам, таким як Bitcoin і Ethereum. Хоча на ринку з'являються нові криптовалюти, вони відповідають високим стандартам безпеки. Цей технологічний прорив відгукнувся на потребу у вдосконаленні традиційних фінансових систем. Спільні зусилля інженерів призвели до створення блокчейну, який вигідно відрізняється від звичайних валют за рядом параметрів, включаючи безпеку, нестабільність курсу, ірeversибельність операцій та прозорість. Технологія блокчейн працює на основі розподілених мереж, що гарантує її надійність. На сьогоднішній день існує ряд інноваційних застосувань блокчейну, таких як NFT-платформи для торгівлі, які постійно розвиваються та відкривають нові горизонти для вивчення і реалізації.

Мета і завдання роботи. Метою кваліфікаційної роботи є розробка ефективного рішення для інвестування на базі технології NFT. Робота націлена на створення комплексного підходу до забезпечення безпеки криптоактивів, надійності транзакцій та доступного користувацького інтерфейсу.

Досягнення поставленої мети передбачає розв'язання наступних завдань:

- Проведення аналізу технології Blockchain;
- Проведення аналізу технології Ethereum;
- Аналіз технології NFT;
- Вивчення та вибір конкретного програмного стеку для реалізації проекту;
- Проектування та налаштування віртуального середовища;
- Проектування та деплой смарт-контракту;
- Розробка веб-інтерфесу для взаємодії з блокчейном;

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Актуальність технології blockchain

Блокчейн – розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшас. Дані захищено від підробки та спотворення. Кожен блок містить часову позначку, геш попереднього блока та дані транзакцій, подані як геш-дерево. Таку розподілену базу даних закладено в основу криптовалюти Bitcoin, де слугує бухгалтерською книгою для всіх операцій. Таку базу називають Блокчейн.

У квітні 2017 року в ЗМІ з'явилася інформація, що Україна планує перевести державні дані у блокчейн з метою: впровадження максимальної безпеки даних; ефективної боротьби з корупцією; усунення рейдерства; прозорості та відкритості даних.

Зростаюча важливість вивчення блокчейну стає очевидною, особливо у контексті глобальних викликів, таких як пандемія COVID-19. Вона акцентує увагу на необхідності надійних даних та стратегій відновлення, спрямованих на цифрову трансформацію та електронну комерцію. Однак технологія блокчейн може викликати сумніви серед тих, хто ще не має досвіду з цим видом цифрових рішень.

Протягом останніх років гостро обговорювалися криптовалюти та технологія блокчейн. Більшість експертів підкреслює переваги блокчейну перед стандартними системами збереження інформації, оскільки вона гарантує безпеку, незмінність та прозорість даних. За думкою світових фахівців, блокчейн незабаром знайде своє місце в різних секторах економіки, і його потенціал революціонізувати традиційні бізнес-процеси на глобальному рівні вважається великим.

Технологія Блокчейн є виразом лібералізму і надає людству як свободу так і самодостатність. Якщо ви спробуєте змінити принцип функціонування блокчейну намаганням врегулювати, блокчейн може стати небезпечним. Ця технологія

вперше застосовувалася людиною або групою людей для створення першої світової криптовалюти – Bitcoin.

Цей проект мав створити революцію для позбавлення від великих посередників фінансового світу, таких як банки та інших фінансові установи. Якщо регулятори спробують відокремити блокчейн від його передбачуваної мети, фундаменталісти блокчейну дадуть у відповідь подальшим розвитком блокчейну, щоб задовольнити свою мету - замінити фінансову систему у тому вигляді, в якому ми її знаємо. Якщо ті, хто займаються регулюванням блокчейну спробують відхилити його від початкової мети, його розробники продовжать розвивати його, щоб домогтися створення фінансової системи відмінною від тієї, яку ми знаємо зараз.

1.2 Аналіз технології Ethereum

Ethereum - це блокчейн із вбудованим в нього комп'ютером. Це основа для створення додатків та організацій у децентралізований, бездозвільний, стійкий до цензури спосіб.

У всесвіті Ethereum існує єдиний канонічний комп'ютер (так звана віртуальна машина Ethereum, або **EVM**), стан якого узгоджується з усіма учасниками мережі Ethereum. Кожен учасник мережі Ethereum (кожна нода Ethereum) зберігає копію стану цього комп'ютера. Крім того, будь-який учасник може транслювати запит на цей комп'ютер для виконання довільних обчислень. Щоразу, коли такий запит транслюється, інші учасники мережі перевіряють, підтверджують і виконують обчислення. Це виконання викликає зміну стану EVM, яка фіксується і поширюється по всій мережі.

Запити на обчислення називаються запитами на транзакції; запис усіх транзакцій і поточного стану EVM зберігається в блокчейні, який, у свою чергу, зберігається і узгоджується всіма вузлами.

Криптографічні механізми гарантують, що після того, як транзакції будуть підтверджені як дійсні і додані до блокчейну, їх неможливо буде підмінити. Ці ж механізми гарантують, що всі транзакції підписуються і виконуються з

відповідними "дозволами" (ніхто не зможе відправити цифрові активи з акаунта Аліси, окрім неї самої).

Ether(ETH) - це рідна криптовалюта Ethereum. Мета ETH - створити ринок для обчислень. Такий ринок забезпечує економічний стимул для учасників перевіряти і виконувати запити на транзакції та надавати обчислювальні ресурси мережі.

Будь-який учасник, який транслює запит на транзакцію, повинен також запропонувати мережі певну кількість ETH в якості винагороди. Мережа присудить цю винагороду тому, хто в кінцевому підсумку виконає роботу з перевірки транзакції, її виконання, фіксації в блокчейні і трансляції в мережу.

Сума випланих ETH відповідає ресурсам, необхідним для виконання обчислень. Ці бенефіти також запобігають навмисному засміченню мережі зловмисниками, які вимагають виконання нескінченних обчислень або інших ресурсоемних скриптів, оскільки ці учасники повинні платити за обчислювальні ресурси.

ETH також використовується для забезпечення криптоекономічної безпеки мережі трьома основними способами:

- 1) він використовується як засіб винагороди валідаторів, які пропонують блоки або повідомляють про нечесну поведінку інших валідаторів;
- 2) валідатори вкладають його в якості застави проти нечесної поведінки - якщо валідатор намагається поводитися нечесно, його ETH може бути знищено;
- 3) він використовується для зважування "голосів" за нові запропоновані блоки, які беруть участь у виборі форків у механізмі консенсусу.

1.3 Розбір поняття смарт-контракт

На практиці учасники не пишуть новий код щоразу, коли хочуть зробити запит на обчислення на EVM. Замість цього розробники додатків завантажують програми (багаторазові фрагменти коду) в стан EVM, а користувачі роблять запити на виконання цих фрагментів коду з різними параметрами. Ми називаємо програми, що завантажуються в мережу і виконуються нею, смарт-контрактами.

На найпростішому рівні смарт-контракт можна уявити як своєрідний торговий автомат: скрипт, який при виклику з певними параметрами виконує певні дії або обчислення, якщо задовольняються певні умови. Наприклад, простий смарт-контракт продавця може створювати і передавати право власності на цифровий актив, якщо абонент надсилає ЕТН конкретному одержувачу.

Будь-який розробник може створити смарт-контракт і оприлюднити його в мережі, використовуючи блокчейн як рівень даних, за певну плату, що сплачується мережі. Потім будь-який користувач може викликати смарт-контракт для виконання його коду, знову ж таки за плату, що сплачується мережі.

Таким чином, за допомогою смарт-контрактів розробники можуть створювати і розгортати як завгодно складні користувацькі додатки і сервіси, такі як: торгові майданчики, фінансові інструменти, ігри тощо.

1.4 Актуальність NFT в сучасному світі.

Термін NFT стає все більш відомим. Геймери активно докупують віртуальні предмети в NFT-іграх, навіть якщо гра ще не вийшла. Цифрові арти, музичні твори та відомі меми у форматі NFT швидко знаходять своїх покупців за великі гроші, здивувавши тих, хто ще не знає про цю концепцію.

NFT — це унікальний токен, кожен з яких є унікальним і не може бути замінений. Це, майже як цифровий папірець у блокчейні, який підтверджує власність конкретного цифрового об'єкта.

NFT відображає різноманітну інформацію про цифрові об'єкти, таку як їх власника, історію переходу власності, авторство та походження. Головна перевага NFT — це можливість перевірити автентичність предмета і зафіксувати право власності.

Технологія NFT була запущена в 2017 році на Ethereum через смарт-контракти. Перша цифрова творча робота, яка стала NFT, належала Бенксі. Оригінал цієї роботи було знищено, а NFT стало його цифровим відображенням. Такий незвичний початок показав потенціал та майбутнє NFT.

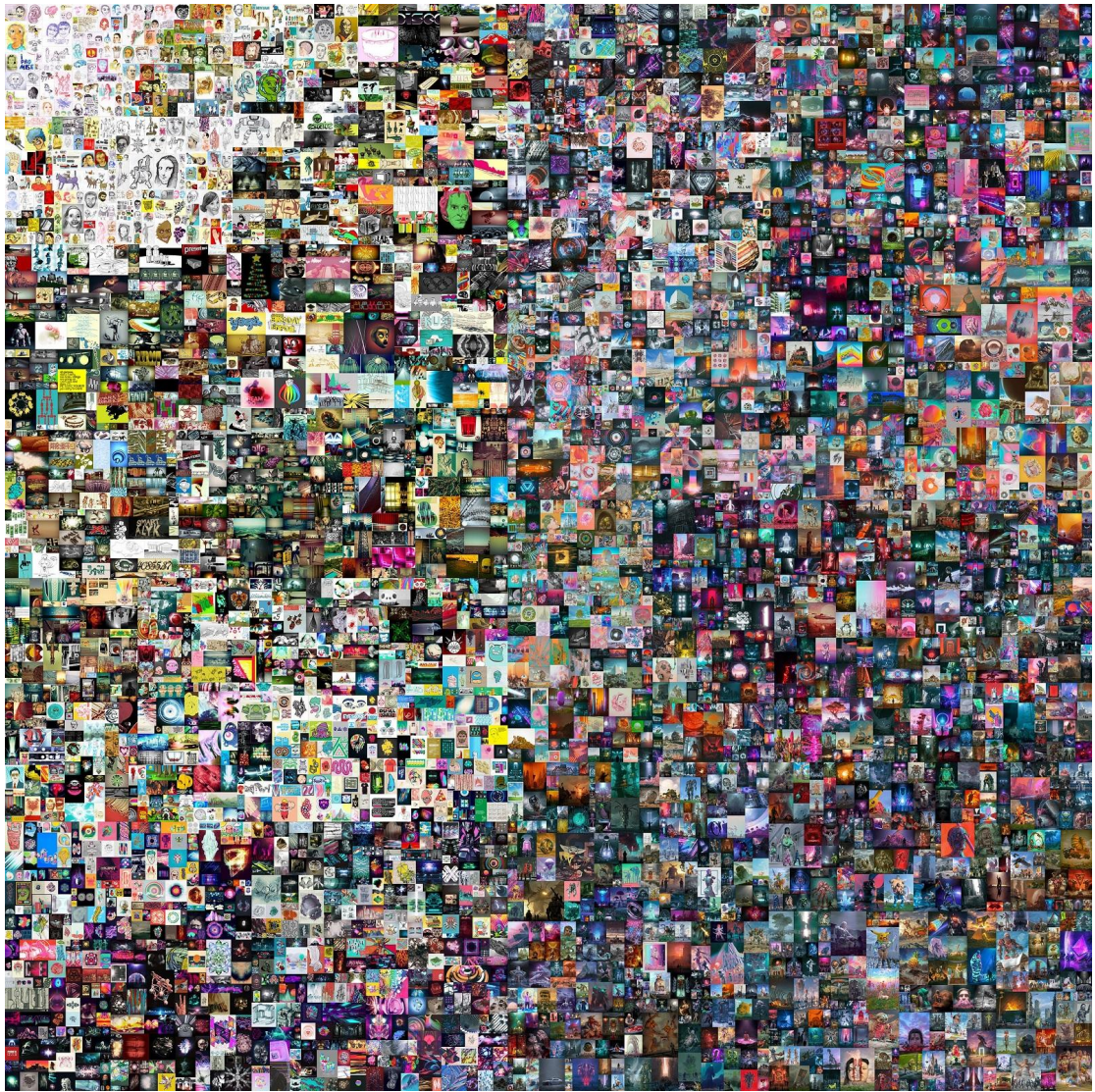


Рис. 1.1. Найдорожча NFT в світі

Зараз NFT найчастіше використовується для підтвердження факту володіння такими цифровими активами, як зображення, відео, меми, музика. В ігровій сфері популярні NFT колекційні предмети та внутрішньоігрові активи. Проте це далеко не все. Нині з допомогою NFT можна токенізувати:

- Твіти (засновник Twitter Джек Дорсі продав NFT-токен свого першого твіту за \$2,9 млн).
- Доменні імена.
- Квитки на будь-які події.
- Право володіння фізичними активами (нерухомість, авто, предмети мистецтва).
- Марки, монети та інші колекційні предмети.

-Ліцензії на книжкові видання, музичні твори та фільми.

Також створення NFT потребує наявність сховища, де можна зберігати медіа-файли, посилання на які вказані в метаданих блоку в блокчейні. Таким сховищем може бути IPFS (InterPlanetary File System) – розподілена система для зберігання та доступу до файлів, веб-сайтів, програм та даних. IPFS знає, як знайти файл (html або медіа), за вмістом, а не за місцем розташування. Таким чином замість одного запиту, наприклад, до одного серверу вікіпедії «що ж таке блокчейн», буде зроблено пошук по багатьом серверам (комп'ютерам), які можуть мати цей файл. За допомогою IPFS, можливо не просто завантажити файли від когось іншого – ваш комп'ютер також допомагає їх розповсюджувати. Коли вашому другу за кілька кварталів потрібна та сама сторінка Вікіпедії, можливо, він отримає її від вас, як і від вашого сусіда чи когось іншого, хто використовує IPFS. IPFS робить це можливим не лише для веб-сторінок, а й для будь-якого файлу, який може зберігати комп'ютер, будь то документ, електронна пошта чи навіть запис бази даних.

Таким чином, завантаження файлу з багатьох місць має наступні переваги:

- Підтримує стійкий Інтернет. Якщо хтось атакує веб-сервери Вікіпедії або інженер у Вікіпедії робить велику помилку, через яку їх сервери загораються, ви все одно можете отримати ті самі веб-сторінки звідкись.

- Ускладнює цензуру вмісту. Оскільки файли на IPFS можуть надходити з багатьох місць.

- Може пришвидшити роботу в Інтернеті, коли ви далеко або від'єднанні. Якщо ви можете отримати файл у когось поруч, а не за сотні чи тисячі кілометрів, ви можете часто отримати його швидше. Це має особливу вагу, якщо ваша спільнота локально пов'язана з мережею, але не має належного зв'язку з ширшим Інтернетом.

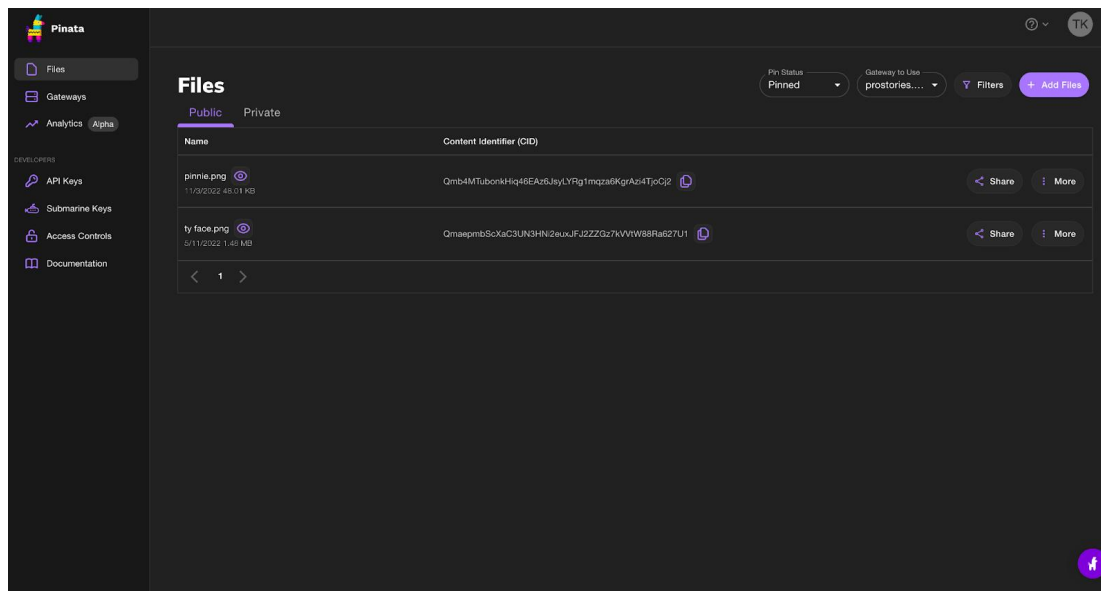


Рис.1.2. Інтерфейс одного з найпопулярніших ipfs сервісів pinata.cloud

1.5 Аналіз ринку NFT маркетплейсів

NFT маркетплейси — це спеціалізовані платформи в сфері криптовалют, де доступні унікальні токени. Чи то токенизовані віртуальні котики на CryptoKitties, чи інші цифрові активи, їх можна знайти на різних платформах. Ці площадки пропонують широкий вибір невзаємозамінних токенів для обміну або купівлі.

Одним з найпопулярніших майданчиків для продажу NFT є **OpenSea**, на якому виставлено на продаж понад 4 мільйони NFT. Платформа пропонує різні типи NFT, включаючи предмети мистецтва, ігрові активи та колекційні речі. OpenSea стягує комісію в розмірі 2,5% з усіх транзакцій, а рейтинг платформи на Trustpilot становить 4,6 з 5. Найбільш продавані товари на OpenSea включають криптовалюту CryptoPunks і Bored Ape Yacht Club NFT.

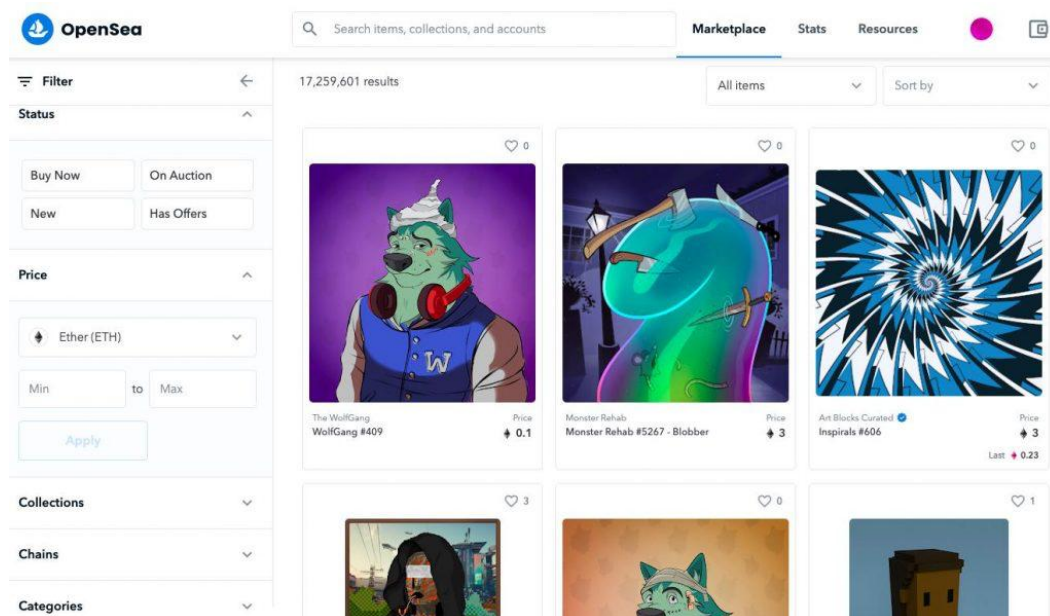


Рис. 1.3. Інтерфес маркетплейсу OpenSea



Рис. 1.4. Колекція Bored Ape Yacht Club NFT

Rarible – ще один популярний NFT-маркетплейс, який дозволяє творцям карбувати свої NFT. Платформа пропонує різноманітні NFT, включаючи твори мистецтва, музику та доменні імена. Rarible стягує 2,5% комісії з усіх транзакцій, а рейтинг платформи на Trustpilot становить 3,8 з 5. Найбільш продавані товари на Rarible включають NFT від таких популярних виконавців, як Граймс і Стів Аокі.

SuperRare — це відбірковий маркетплейс NFT, спрямований на мистецтво. Тут можна знайти унікальні та ексклюзивні твори від різних талановитих митців. За кожну операцію платформа бере 3% комісії. На платформі Trustpilot SuperRare отримує високий рейтинг 4,8 з 5. Між найпопулярніших робіт на SuperRare входять твори від відомих художників, таких як Пак та Тревор Джонс.

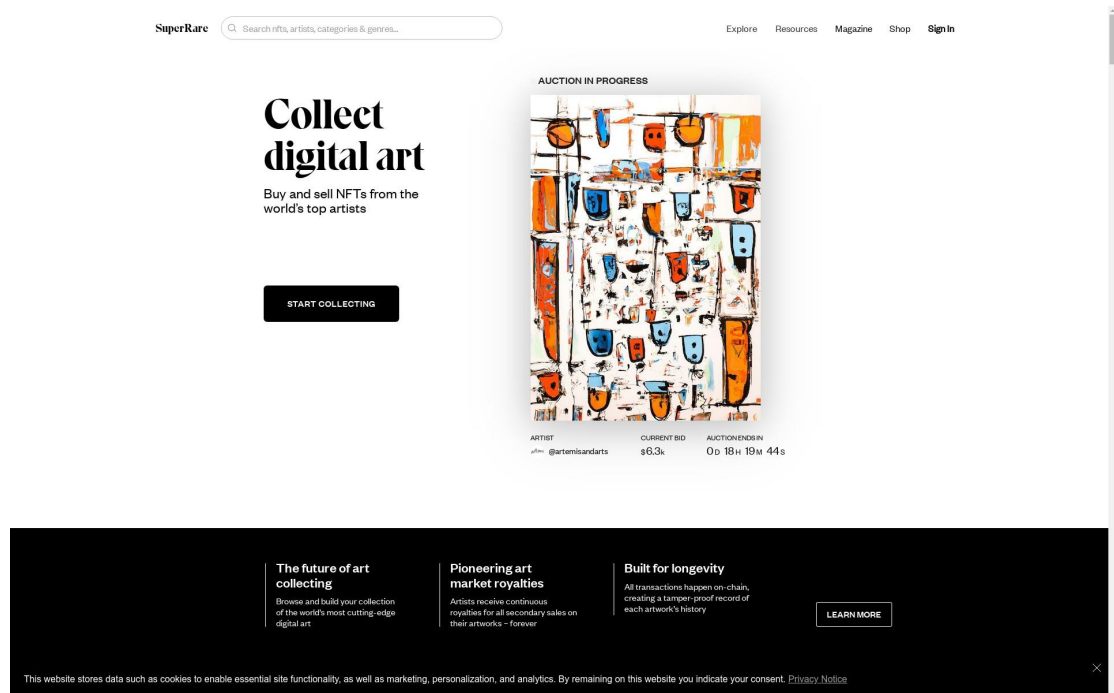


Рис. 1.5. SuperRare веб інтерфейс

1.6 Постановка завдання на розробку

Метою кваліфікаційної роботи є розробка зручного в використанні та зрозумілого веб інтерфейсу для безпечного інвестування криптоактивів за допомогою NFT. Проект являтиме собою веб рішення, яке буде взаємодіяти зі смарт контрактом.

Розробка складатиметься з наступних кроків:

- вибір необхідних інструментів
- налаштування проекту, вивчення необхідних бібліотек
- розгортання та налаштування локальної блокчейн ноди
- розробка смарт-контракту

- реалізація користувацького веб інтерфейсу
- налаштування взаємодії користувацького веб інтерфейсу та смарт-контракту

Висновки до 1-го розділу:

Проаналізовано актуальну інформацію стосовно розвитку технології та можливостей, розібрано найпопулярніші варіанти реалізації функціоналу який надає Ethereum.

В результаті аналізу специфіки NFT смарт-контрактів було сформоване уявлення про принцип роботи, взаємозв'язки та сильні та слабкі сторони їх складових.

Визначено особливості розробки WEB3 орієнтованих інтерфейсів та основні напрямки дослідження.

РОЗДІЛ 2 ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ. РЕАЛІЗАЦІЯ ПРОЄКТУ

2.1 Вибір технологій

2.1.1 Next.js

Next.js — це вебфреймворк з відкритим вихідним кодом, створений приватною компанією Vercel, що дозволяє розробляти веб-додатки на основі React із рендерингом на стороні сервера та генерацією статичних веб-сайтів.

Документація React згадує Next.js серед «Рекомендованих наборів інструментів» і рекомендує його розробникам під час «створення серверного веб-сайту за допомогою Node.js». У той час як традиційні сайти React можуть відтворювати свій вміст лише в браузері на стороні клієнта, Next.js розширює цю функціональність, додаючи можливість генерації веб-сайту на стороні сервера.

Авторські права та торгові марки для Next.js належать компанії Vercel, яка підтримує та очолює розробку фреймворку як програмного забезпечення з відкритим вихідним кодом.

Для розробки було встановлено і налаштовано **Visual Studio Code** - це редактор початкового коду, створений Microsoft із Electron Framework для Windows, Linux і macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають функціональність.

В опитуванні розробників Stack Overflow 2022 серед 71 010 респондентів Visual Studio Code назвали найпопулярнішим інструментом середовища розробника, при цьому 74,48% повідомили, що вони ним користуються.

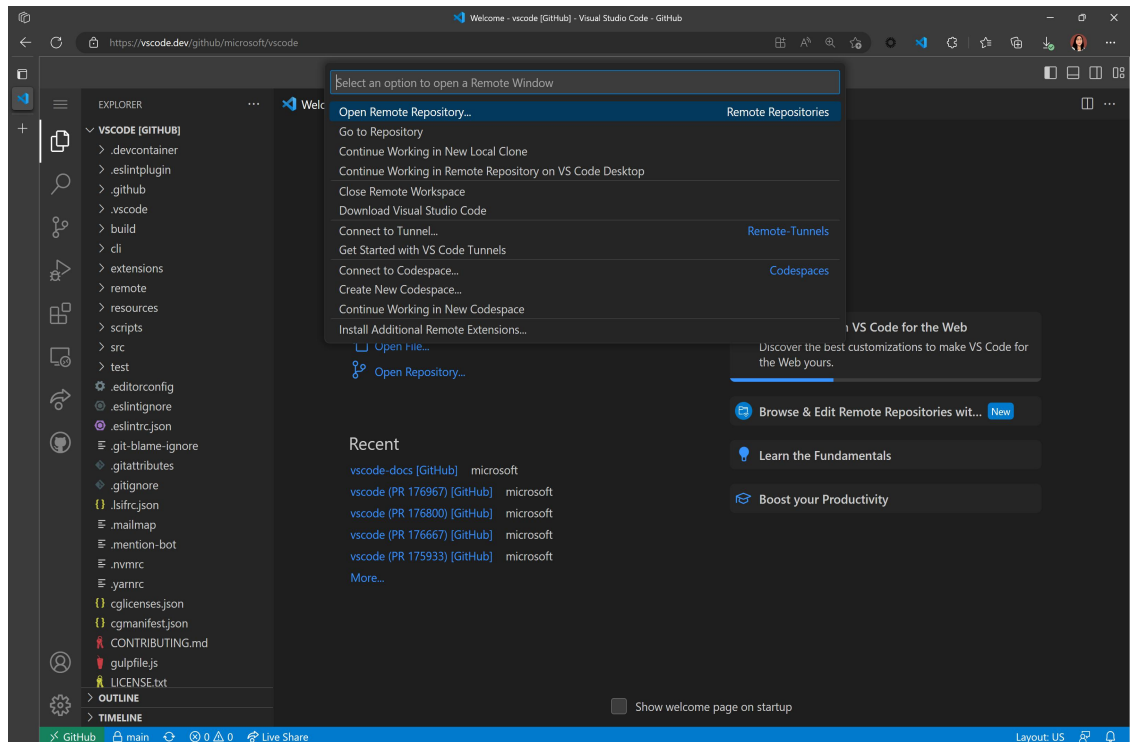


Рис. 2.1.1. Інтерфейс Visual Studio Code

Було проведено ініціалізацію нового проекту, створено файл `package.json` куди додали основну конфігурацію проекту.

`package.json`:

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "dev:blockchain": "truffle development",
    "lint": "next lint",
    "genContractType": "abi-types-generator './public/contracts/NftMarket.json' --
output='./types' --name=nftMarketContract --provider=ethers_v5"
  },
  "dependencies": {
    "@apollo/client": "^3.8.4",
```

"@emotion/react": "^11.11.1",
"@emotion/styled": "^11.11.0",
"@metamask/providers": "^14.0.2",
"@mui/material": "^5.14.10",
"@openzeppelin/contracts": "^4.5.0",
"@tabler/icons-react": "^2.35.0",
"@truffle/hdwallet-provider": "^2.1.15",
"@types/node": "20.5.1",
"@types/react": "18.2.20",
"@types/react-dom": "18.2.7",
"autoprefixer": "10.4.15",
"axios": "^1.3.6",
"deepmerge": "^4.3.1",
"ethereum-abi-types-generator": "^1.3.4",
"ethers": "5.6.2",
"framer-motion": "^10.16.4",
"lodash": "^4.17.21",
"next": "13.4.19",
"next-auth": "^4.22.3",
"postcss": "8.4.28",
"react": "18.2.0",
"react-country-region-selector": "^3.6.1",
"react-dom": "18.2.0",
"react-icons": "^4.10.1",
"react-phone-input-2": "^2.15.1",
"react-toastify": "^9.1.3",
"solc": "^0.8.20",
"swr": "^2.2.4",
"tailwindcss": "3.3.3",
"truffle": "^5.11.5",

```
"typescript": "5.1.6",
},
"devDependencies": {
"@types/lodash": "^4.14.199",
"@types/node": "^20",
"@types/react": "^18",
"@types/react-dom": "^18",
"autoprefixer": "^10.4.16",
"eslint": "^8",
"eslint-config-next": "14.0.3",
"postcss": "^8.4.31",
"tailwindcss": "^3.3.5",
"typescript": "^5"
}
}
```

2.1.2 Metamask

MetaMask - це популярне і визнане розширення для браузера, яке функціонує як криптогаманець, який підключається до блокчейну Ethereum. MetaMask дозволяє користувачам взаємодіяти з екосистемою Ethereum, в якій розміщено величезний всесвіт децентралізованих додатків (Dapps), без необхідності завантажувати весь блокчейн на свій пристрій. Таким чином, це одне з найкращих рішень гаманця Ethereum для легкого доступу децентралізованих бірж (DEX), ігрових платформ, сайтів азартних ігор та багатьох інших програм.

Гаманець сумісний з найбільш поширеними браузерами, такими як Chrome, Firefox, Brave і Microsoft Edge. Крім зберігання власної валюти Ethereum, ETH, MetaMask також містить токени, створені за стандартами протоколу ERC-20 і ERC-721.

MetaMask був заснований Аароном Девісом (Aaron Davis) та блокчейн компанією ConsenSys.

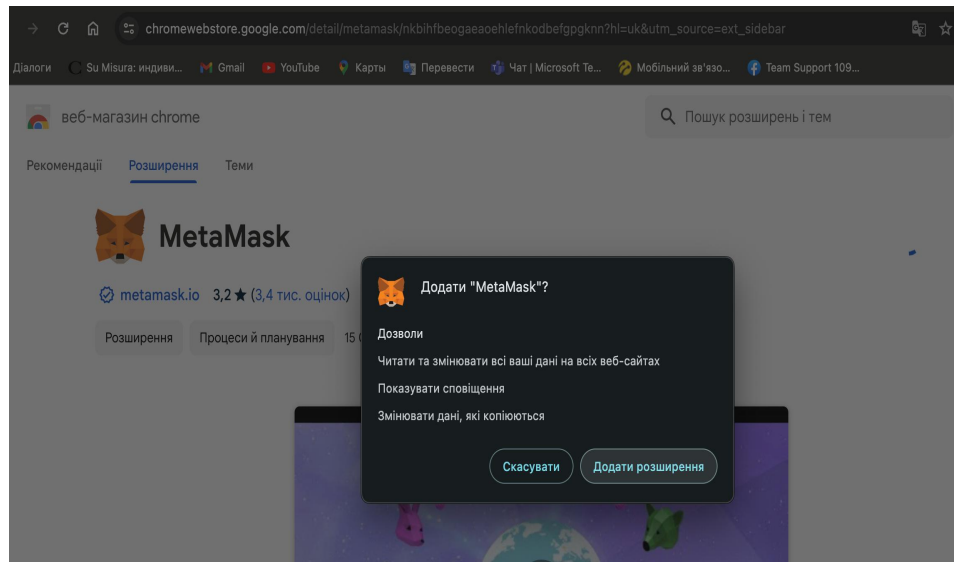


Рис. 2.1.2 Встановлення офіційного розширення MetaMask

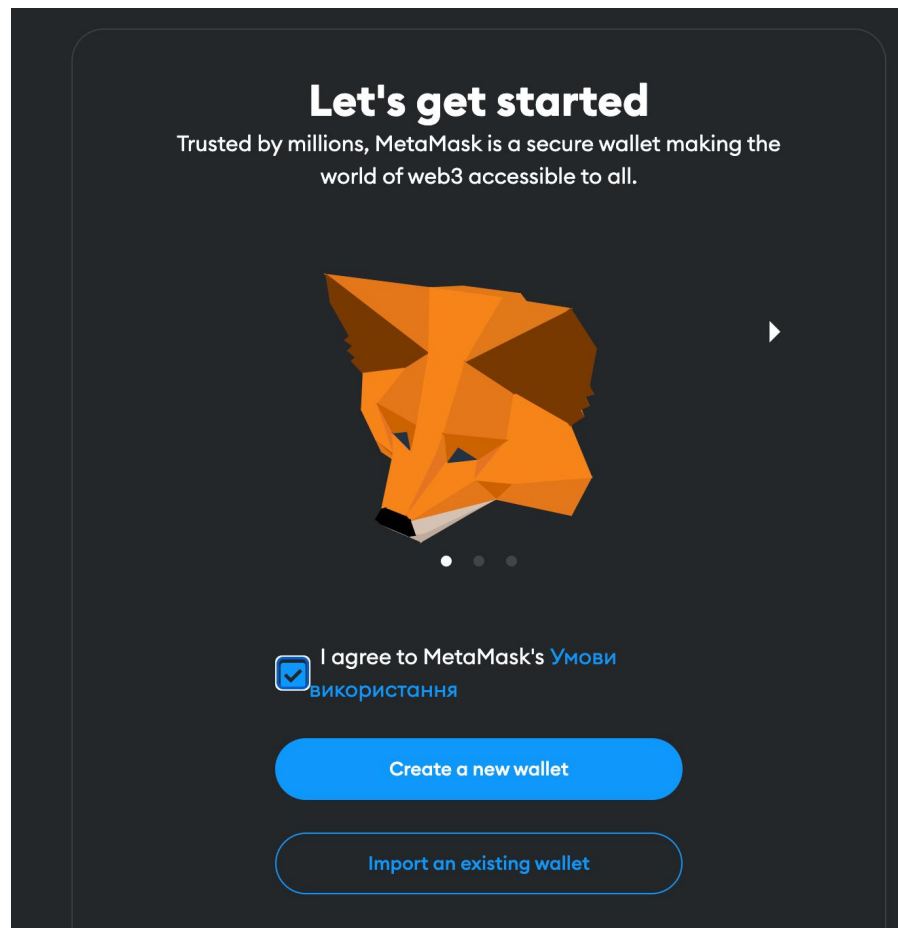


Рис. 2.1.3 Створення гаманця MetaMask

За допомогою ресурсу lastpass.com було згенеровано новий надійний пароль для доступу до додатку MetaMask.

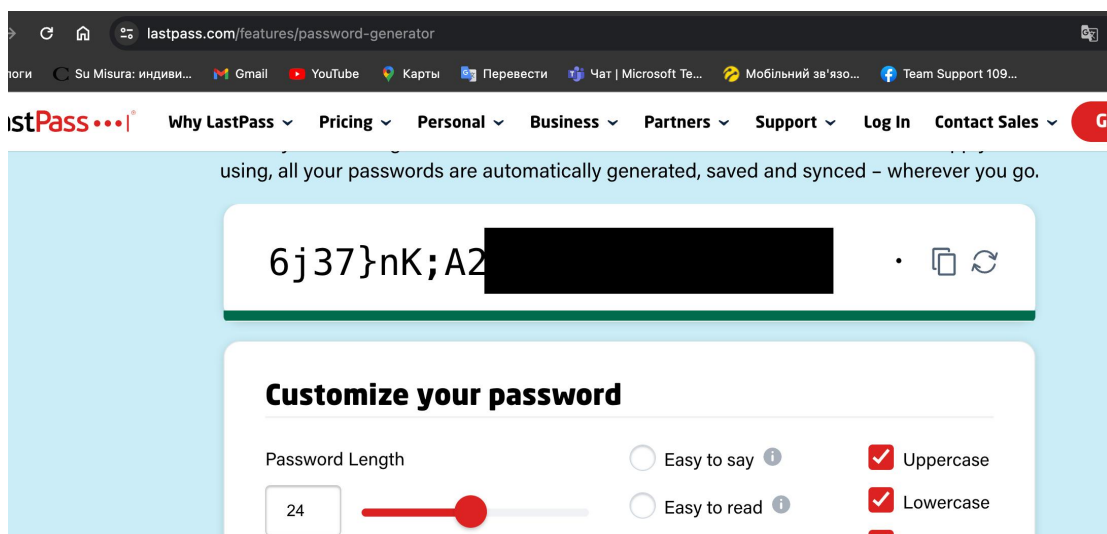


Рис. 2.1.4 Генерація надійного пароля

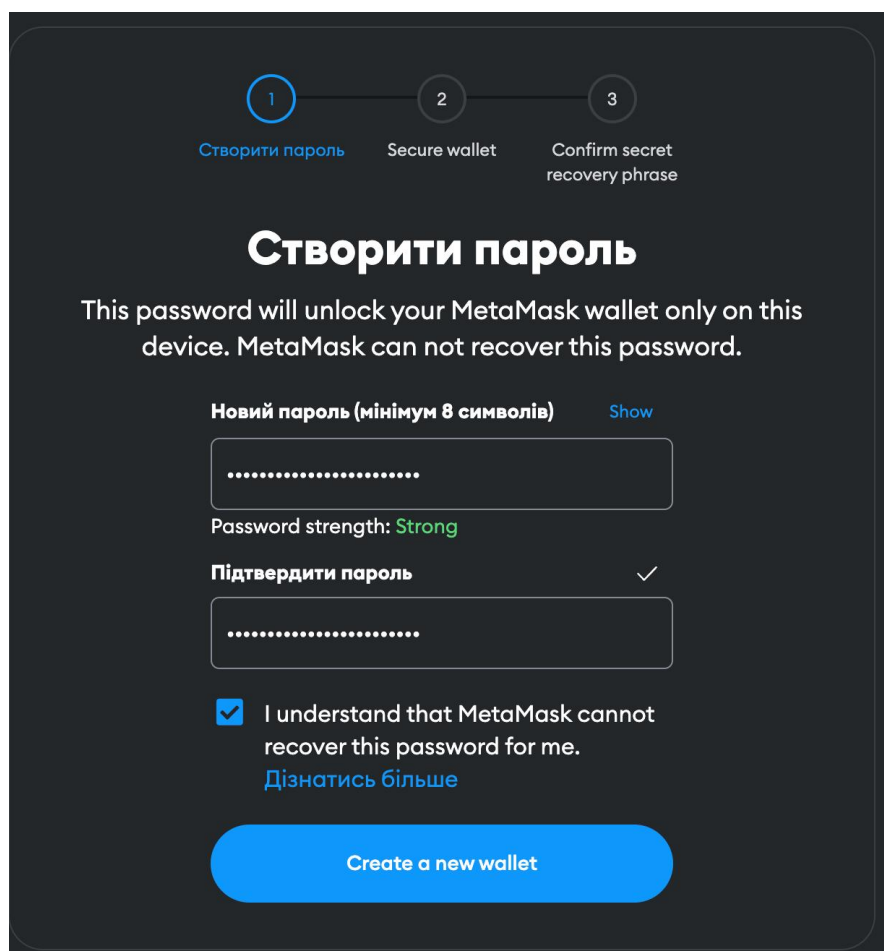


Рис. 2.1.5 Створення нового паролю MetaMask

Створити пароль Secure wallet Confirm secret recovery phrase

Write down your Secret Recovery Phrase

Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

Чайові:

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

1.		2.	run	3.	
4.	client	5.		6.	fly
7.		8.	proof	9.	
10.	dad	11.		12.	seed

[Hide seed phrase](#) [Копіювати в буфер](#)

Далі

Рис. 2.1.6 Отримана фраза відновлення гаманця

1 2 3
Створити пароль Secure wallet Confirm secret recovery phrase

Confirm Secret Recovery Phrase

Confirm Secret Recovery Phrase

1.		2.	run	3.	
4.		5.		6.	
7.	bench	8.		9.	blind
10.	dad	11.		12.	

Підтвердити

Рис. 2.1.7 Підтвердження фрази відновлення

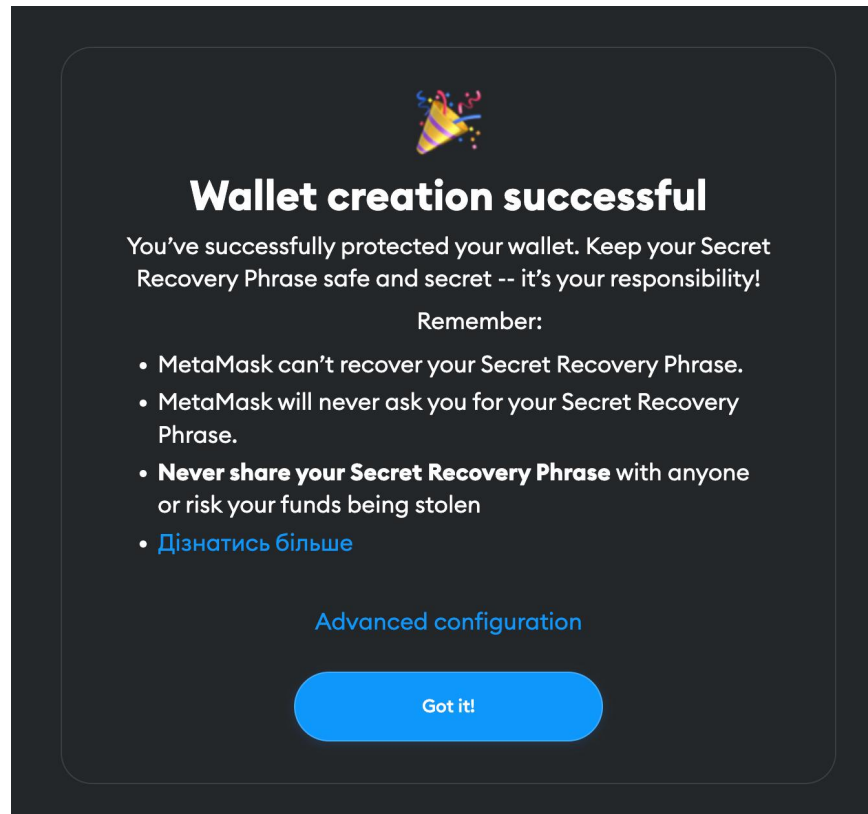


Рис. 2.1.9 Успішно створено новий гаманець

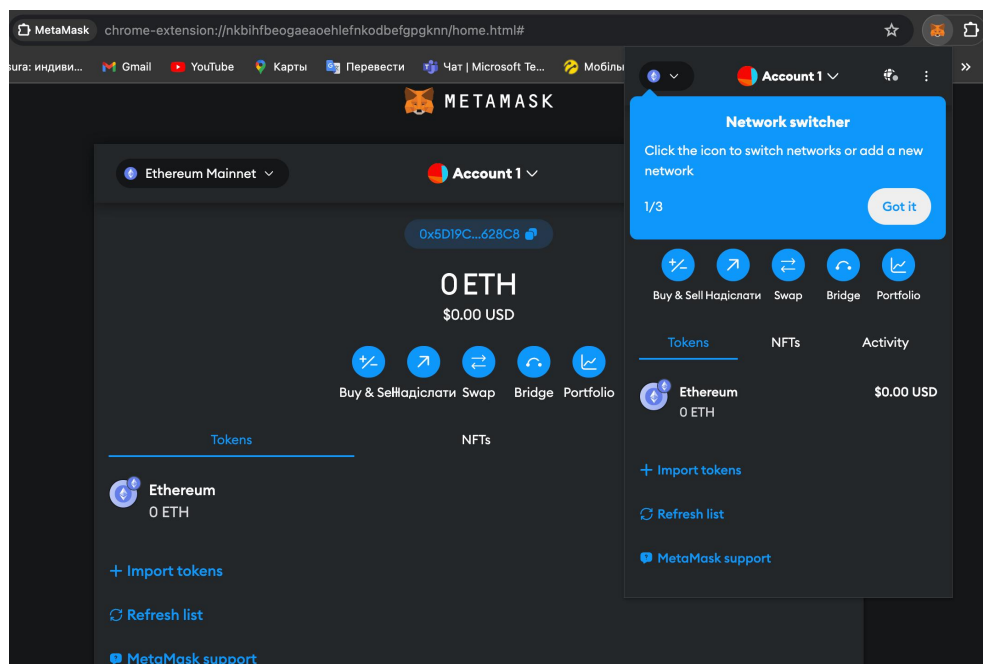


Рис. 2.1.10 Основний інтерфейс гаманця MetaMask

2.1.3 Solidity

Solidity — це об'єктно-орієнтована мова програмування високого рівня для написання та виконання смарт-контрактів на блокчейні. Найбільш відомий своїм зв'язком з домівкою DeFi, Ethereum, Solidity був створений розробниками Ethereum для створення смарт-контрактів, які працюють із віртуальною машиною Ethereum у мережі.

У просторі розробників web3 Solidity є однією з найпоширеніших мов програмування, хоча вона містить елементи з інших популярних мов, таких як Javascript, C++ і Python. Solidity — це мова зі статистичною типізацією, яка підтримує успадкування, складні визначені користувачем типи та бібліотеки — елементи, необхідні для функціонування смарт-контрактів. Solidity є популярним вибором серед розробників смарт-контрактів, оскільки він простий у використанні. Природа мови програмування (високого рівня) замінює введення бітового коду (одиниць і нулів) комбінацією літер і цифр.

Незважаючи на поточну репутацію однієї з найпоширеніших мов програмування, Solidity має скромний початок, який сягає далекого 2014 року. У 2014 році Гевін Вудс, Крістіан Райтвіснер та інші розробники Ethereum Foundation створили Solidity для підтримки розвитку майбутньої мережі Ethereum. Розробники створили Solidity як мову, яку розробники знайшли простою у використанні, виразною та сумісною з віртуальною машиною Ethereum, середовищем виконання, яке є центральним для виконання смарт-контрактів.

Solidity натхненний такими існуючими мовами, як Python, C++ і JavaScript. Тим не менш, його оригінальність походить від унікальних концепцій і синтаксису, вбудованих спеціально для роботи зі смарт-контрактами, змінними стану, модифікаторами та подіями.

Solidity в основному використовується для написання смарт-контрактів. Смарт-контракти є основою мережі Ethereum і стосуються програм, які працюють на віртуальній машині Ethereum (EVM) і виконують численні функції, в основному пов'язані з автоматизацією транзакцій у мережі. EVM працює як зберігач записів для «глобального комп'ютера» Ethereum, що означає, що EVM

підтримує глобальний стан мережі, включаючи адреси, баланси рахунків і запаси токенів.

Контракти Solidity компілюються в байт-код і зчитуються EVM, що означає, що теоретично розробники можуть писати смарт-контракти Ethereum безпосередньо за допомогою байт-коду. Однак запис безпосередньо за допомогою байт-коду майже не використовується, оскільки він є більш технічним і схильним до помилок. Таким чином, використання мов високого рівня, таких як Solidity, дозволяє розробникам зосередитися на логіці та функціональності коду замість того, щоб мати справу зі складнощами написання в байт-коді. У результаті розробники можуть кодувати ефективніше та більше зосереджуватися на створенні розумних контрактів, які легше читати, писати та підтримувати.

Solidity підлягає періодичному оновленню версій, які випускаються на Github. Смарт-контракти незмінні, тобто після їх розгортання в ланцюжку їх неможливо скасувати. Оскільки сама Solidity постійно розвивається, незмінність смарт-контрактів впливає на неї, оскільки будь-які помилки чи недоліки в коді контракту не можна легко виправити після того, як він активний. Причина в тому, що кожна версія має відмінну поведінку та особливості, які можуть впливати на функціональність контракту та те, як він взаємодіє з іншими елементами в екосистемі Ethereum.

Останнім часом розробники віддають перевагу використанню проксі-контрактів і контрактів з можливістю оновлення. Проксі-контракти дозволяють виконувати код іншого контракту. Проксі-сервери діють як посередники, перенаправляючи дзвінки та транзакції до передбаченого контракту, одночасно надаючи додаткові функції, такі як контроль доступу або журналювання. Контракти з можливістю оновлення по суті включають смарт-контракти, які можна оновлювати без втрати початкового стану.

Слід зазначити, що версія Solidity, яка використовувалася під час створення смарт-контракту, також є важливою, оскільки це визначатиме поведінку та функції смарт-контракту. Кожен рядок вихідного коду кожного смарт-контракту Solidity, яким би простим він не був, має починатися з оголошення того, яку

версію компілятора Solidity слід використовувати для цього конкретного фрагмента коду. Ця декларація відома як «прагма версії».

2.1.4 Truffle

Truffle - це середовище розробки світового класу, фреймворк для тестування та конвеєр активів для блокчейнів з використанням віртуальної машини Ethereum (EVM), що має на меті полегшити життя розробника.

Truffle вважається найпопулярнішим інструментом для розробки блокчейн-додатків з більш ніж 1,5 мільйонами завантажень за весь час існування. Truffle підтримує розробників протягом усього життєвого циклу їхніх проектів, незалежно від того, чи хочуть вони використовувати Ethereum, Hyperledger, Quorum або одну з постійно зростаючого списку інших підтримуваних платформ. У поєднанні з Ganache, персональним блокчейном, та Drizzle, інтерфейсним набором для розробки dApp, повний набір інструментів Truffle обіцяє стати комплексною платформою для розробки dApp.

Вбудована компіляція смарт-контрактів, зв'язування, розгортання та управління бінарними файлами.

Основні переваги:

- Автоматизоване тестування контрактів для швидкої розробки.
- Фреймворк для розгортання та міграції, що розширюється за допомогою сценаріїв.
- Управління мережею для розгортання в будь-якій кількості публічних і приватних мереж.
- Управління пакетами за допомогою EthPM та NPM, використовуючи стандарт ERC190.
- Інтерактивна консоль для прямого контрактного спілкування.
- Конвеєр збірки з підтримкою тісної інтеграції.
- Зовнішній запуск скриптів, який виконує скрипти в середовищі Truffle.

Розроблено конфігураційний файл `truffle-config.js`, додано дві мережі: локальна та тестова нода **Sepolia**.

truffle-config.js:

```
const HDWalletProvider = require("@truffle/hdwallet-provider");
const mnemonic = "";

module.exports = {
  contracts_build_directory: "./public/contracts",
  networks: {
    development: {
      host: "127.0.0.1", // Localhost (default: none)
      port: 9545, // Standard Ethereum port (default: none)
      network_id: "*", // Any network (default: none)
    },
    sepolia: {
      provider: function () {
        return new HDWalletProvider(mnemonic, 'https://wild-maximum-
wind.ethereum-sepolia.quiknode.pro/12497e6874c137f4eb46dc28f7e574c54f51925a')
      },
      network_id: 4868603,
      gas: 4000000, // Adjust the gas limit as per your requirements
      gasPrice: 10000000000, // Set the gas price to an appropriate value
      confirmations: 2, // Set the number of confirmations needed for a transaction
      timeoutBlocks: 200, // Set the timeout for transactions
      skipDryRun: true // Skip the dry run option
    },
    compilers: {
      solc: {
        version: "0.8.13",
      }
    }
  }
}
```

};

2.1.5 Ganache

Ganache - це персональний блокчейн для швидкої розробки розподілених додатків на Ethereum та Filecoin. Ви можете використовувати Ganache протягом усього циклу розробки, що дозволяє вам розробляти, розгортати та тестувати ваші dApps у безпечному та детермінованому середовищі.

Ganache поставляється в двох варіантах: інтерфейс користувача і CLI. Ganache UI - це десктопний додаток, що підтримує технології Ethereum та Filecoin. Для розробки Ethereum доступний наш більш надійний інструмент командного рядка ganache. Він пропонує:

- console.log в Solidity
- Форкінг Mainnet і тестової мережі з нульовою конфігурацією
- Розгалужуйте будь-яку мережу Ethereum, не чекаючи синхронізації
- Підтримка Ethereum JSON-RPC
- Знімок/відновлення стану
- Блокування миттєво, на вимогу або з певним інтервалом
- Перемотування часу вперед
- Імітуйте будь-який обліковий запис (приватні ключі не потрібні!)
- Слухає запити JSON-RPC 2.0 через HTTP/WebSockets
- Програмне використання в Node.js
- Відкладені транзакції

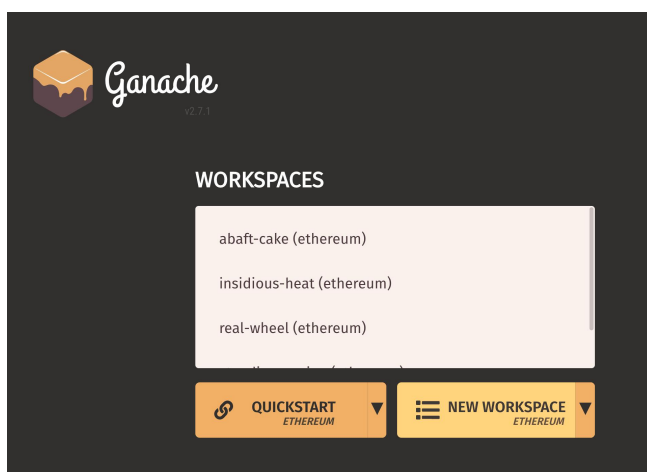


Рис. 2.1.11 Початковий інтерфейс програми Ganache

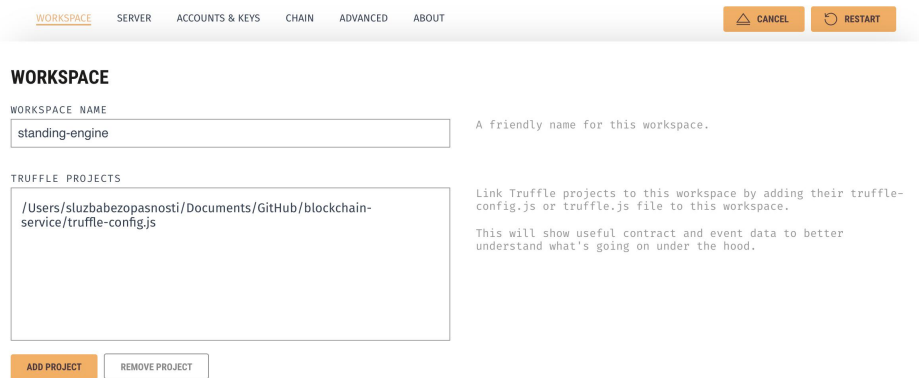


Рис. 2.1.12 Розгортаємо робоче оточення з вже підготовленого truffle-config.js

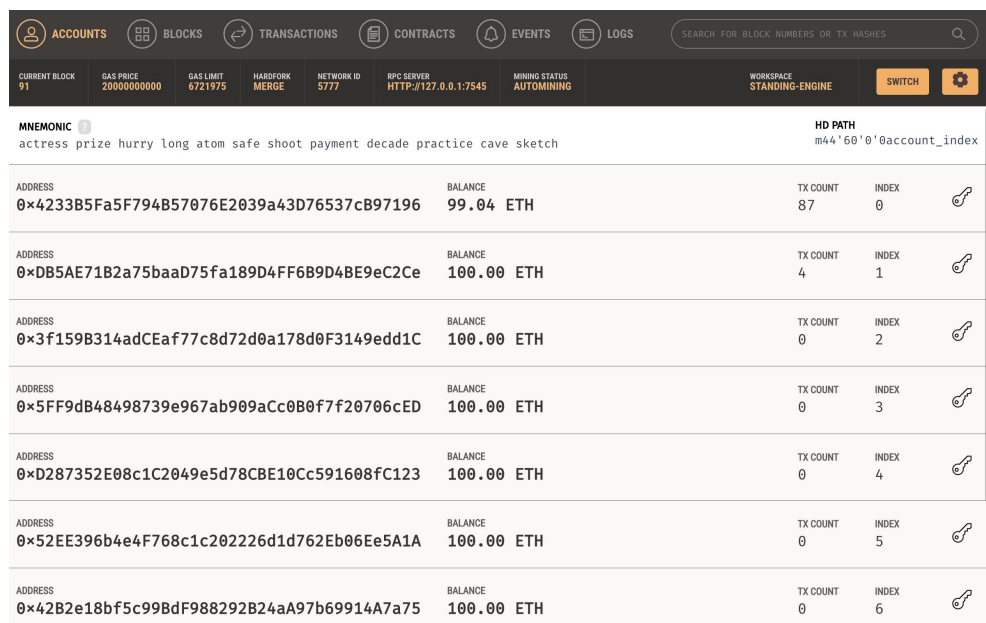


Рис. 2.1.12 Робоча область локального блокчейну

Висновки розділу 2.1: Було проведено детальний аналіз необхідних для реалізації інструментів, проведено дослідження інтерфейсу програмного забезпечення. Для реалізації проєкту будуть використані такі технології:

- Next.JS
- MetaMask
- Solidity
- Truffle

2.2 Розробка смарт-контракту

Було проведено детальний аналіз та визначено перелік основного функціоналу смарт-контракту(Таб 2.1).

Таб 2.1

Назва функції	Опис
getNftItem	Отримуємо інформацію про нфт
listedItemsCount	Отримуємо кількість NFT на лістингу
totalSupply	Отримуємо кількість всіх NFT
tokenByIndex	Отримуємо токен по індексу в масиві
tokenOfOwnerByIndex	Отримуємо всі токени власника по індексу
getOwnedNfts	Отримуємо всі нфт власника
burnToken	Знищити токен
mintToken	Мінт нової нфт
_createNftItem	Створюємо об'єкт NFT
_beforeTokenTransfer	Зміна власника токена
_addTokenToAllTokensEnumeration	Додає новий токен до всіх зв'язаних сховищ
_addTokenToOwnerEnumeration	Присвоєння токена власнику
_removeTokenFromOwnerEnumeration	Видаляє токен від власника
_removeTokenFromAllTokensEnumeration	Видаляє токен з усіх зв'язаних сховищ

Після формування таблиці з переліком функціоналу, було розроблено смарт-контракт на мові Solidity, який забезпечує безпечне списання коштів і створення NFT, дає можливість перегляди власні NFT, інвестувати в нові, видаляти та передавати токени.

За основу смарт-контракту було взято стандарт ERC721, який фактично започаткував феномен NFT. Цей стандарт був першим у своєму роді. Хоча незамінні токени існували вже давно, безпосередньо стандарт ERC 721 відкрив ворота для масової популярності NFT. ERC-721 відіграв важливу роль у виставі унікальних NFT громадськості за допомогою CryptoKitties.

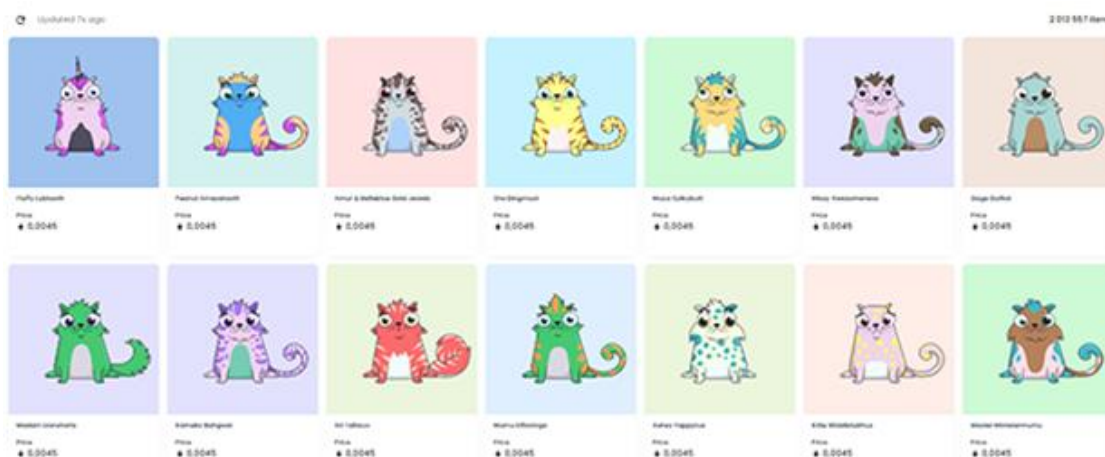


Рис. 2.2.1 CryptoKitties – NFT-гра, присвячена розведенню однойменних колекційних істот

Розробник CryptoKitties, компанія Dapper Labs, у 2017 році запропонувала ERC-721 для вдосконалення Ethereum або EIP (пропозиції щодо покращення Ethereum). CryptoKitties – це набір унікальних, випадково згенерованих кошенят – цифрових персонажів для колекціонування. Слід зазначити, що CryptoKitties – повністю унікальні, їх неможливо повторити. Крім того, історія транзакцій дає чітке уявлення про володіння кошеням з моменту його створення.

Важливо відзначити, що токени ERC-721 – це не тільки повна унікальність, а й такі додаткові характеристики:

- гнучкість переказу NFT між рахунками поруч із можливістю обміну NFT інші валюти;

- прості можливості виявлення власників конкретного активу;
- ефективне визначення загального обсягу пропозиції для будь-якого набору NFT у мережі.

ERC-721 фактично став головним чинником підвищення популярності NFT. Багато токенів, заснованих на цьому стандарті, домінують у криптовалютному ландшафті. Це художні твори, аватари, проєкти PFP – групи аватарів та дрібне володіння віртуальними земельними ділянками.

Код Смар-контракту:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;
import
"@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
contract Crossnetics is ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _listedItems;
    Counters.Counter private _tokenIds;

    // All Token Ids in array
    uint256[] private _allNfts;

    mapping(string => bool) private _usedTokenURIs;
    mapping(uint => NFTItem) private _idToNftItem;

    mapping(address => mapping(uint => uint)) private _ownedTokens;
    mapping(uint => uint) private _idToOwnedIndex;

    mapping(uint => uint) private _idToNftIndex;
```

```
struct NFTItem {  
    uint tokenId;  
    uint price;  
    address creator;  
    bool isListed;  
}
```

```
event NFTItemCreated (  
    uint tokenId,  
    uint price,  
    address creator,  
    bool isListed  
);
```

```
uint public listingPrice = 0.025 ether;
```

```
constructor() ERC721("CreaturesNFT", "CNFT") {}
```

```
function setListingPrice(uint newPrice) external onlyOwner {  
    require(newPrice > 0, "Price must be at least 1 wei");  
    listingPrice = newPrice;  
}
```

```
function getNftItem(uint tokenId) public view returns (NFTItem memory) {  
    return _idToNftItem[tokenId];  
}
```

```
function listedItemsCount() public view returns (uint) {  
    return _listedItems.current();  
}
```

```

function tokenURIExists(string memory tokenURI) public view returns (bool) {
    return _usedTokenURIs[tokenURI] == true;
}

```

```

function totalSupply() public view returns (uint) {
    return _allNfts.length;
}

```

```

function tokenByIndex(uint index) public view returns (uint) {
    require(index < totalSupply(), "Index out of bounds");
    return _allNfts[index];
}

```

```

function tokenOfOwnerByIndex(address owner, uint index) public view returns
(uint) {
    require(index < ERC721.balanceOf(owner), "Index out of bounds");
    return _ownedTokens[owner][index];
}

```

```

function getOwnedNfts() public view returns (NFTItem[] memory) {
    uint ownedItemsCount = ERC721.balanceOf(msg.sender);
    NFTItem[] memory items = new NFTItem[](ownedItemsCount);

```

```

    for (uint i = 0; i < ownedItemsCount; i++) {
        uint tokenId = tokenOfOwnerByIndex(msg.sender, i);
        NFTItem storage item = _idToNftItem[tokenId];
        items[i] = item;
    }
}

```

```

return items;
}

function burnToken(uint tokenId) public {
    _burn(tokenId);
}

function mintToken(string memory tokenURI, uint price) public payable returns
(uint) {
    // require(!tokenURIExists(tokenURI), "Token URI already exists");
    // require(msg.value == listingPrice, "Price must be equal to listing price");
    _tokenIds.increment();
    _listedItems.increment();

    uint newTokenId = _tokenIds.current();
    _safeMint(msg.sender, newTokenId);
    _setTokenURI(newTokenId, tokenURI);
    _createNftItem(newTokenId, price);
    _usedTokenURIs[tokenURI] = true;

    return newTokenId;
}

function _createNftItem(uint tokenId, uint price) private {
    require(price > 0, "Price must be at least 1wei");

    _idToNftItem[tokenId] = NFTItem(
        tokenId,
        price,
        msg.sender,

```

```

    true
);

emit NFTItemCreated(tokenId, price, msg.sender, true);
}

function _beforeTokenTransfer(address from, address to, uint tokenId) internal
virtual override {
    super._beforeTokenTransfer(from, to, tokenId);

    if (from == address(0)) {
        _addTokenToAllTokensEnumeration(tokenId);
    } else if (from != to) {
        _removeTokenFromOwnerEnumeration(from, tokenId);
    }

    if (to == address(0)) {
        _removeTokenFromAllTokensEnumeration(tokenId);
    } else if (to != from) {
        _addTokenToOwnerEnumeration(to, tokenId);
    }
}

function _addTokenToAllTokensEnumeration(uint tokenId) private {
    _idToNftIndex[tokenId] = _allNfts.length;
    _allNfts.push(tokenId);
}

function _addTokenToOwnerEnumeration(address to, uint tokenId) private {
    uint lenght = ERC721.balanceOf(to);

```

```

    _ownedTokens[to][length] = tokenId;
    _idToOwnedIndex[tokenId] = length;
}

```

```

function _removeTokenFromOwnerEnumeration(address from, uint tokenId)
private {

```

```

    uint lastTokenIndex = ERC721.balanceOf(from) - 1;
    uint tokenIndex = _idToOwnedIndex[tokenId];

```

```

    if (tokenIndex != lastTokenIndex) {
        uint lastTokenId = _ownedTokens[from][lastTokenIndex];
        _ownedTokens[from][tokenId] = lastTokenId;
        _idToOwnedIndex[lastTokenId] = tokenIndex;
    }

```

```

    delete _idToOwnedIndex[tokenId];
    delete _ownedTokens[from][lastTokenIndex];
}

```

```

function _removeTokenFromAllTokensEnumeration(uint tokenId) private {

```

```

    uint lastTokenIndex = _allNfts.length - 1;
    uint tokenIndex = _idToNftIndex[tokenId];
    uint lastTokenId = _allNfts[lastTokenIndex];

```

```

    _allNfts[tokenIndex] = lastTokenId;
    _idToNftIndex[lastTokenId] = tokenIndex;

```

```

    delete _idToNftIndex[tokenId];
    _allNfts.pop();
}

```

```
}
```

2.3 Компіляція та деплой смарт контракту

Для того щоб кожен, хто під'єднаний до ноди Ethereum міг використовувати наш контракт і замінити свій токен, контракт має бути скомпільований(приведений до json формату) та задеплойований до необхідної ноди. В цьому нам допоможе вище проаналізоване ПЗ, а саме Truffle, за допомогою команди **truffle compile** ми компілюємо контракт і зберігаємо його до вказаної в конфігурації Truffle папки(/public/contracts).

```
sluzbabezopasnosti@MBP-Sluzba frontend % truffle compile
Compiling your contracts...
=====
> Compiling ./contracts/NftMarket.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compiling @openzeppelin/contracts/utils/Counters.sol
> Compiling @openzeppelin/contracts/utils/Strings.sol
> Compiling @openzeppelin/contracts/utils/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/utils/introspection/IERC165.sol
> Artifacts written to /Users/sluzbabezopasnosti/Documents/GitHub/StagePoint/frontend/public/contracts
> Compiled successfully using:
   - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang
```

Рис. 2.3.1 Результат виконання команди truffle compile

```
{
  "contractName": "NftMarket",
  "abi": [
    {
      "inputs": [],
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "anonymous": false,
      "inputs": [
        {
          "indexed": true,
          "internalType": "address",
          "name": "owner",
          "type": "address"
        },
        {
          "indexed": true,
          "internalType": "address",
          "name": "approved",
          "type": "address"
        },
        {
          "indexed": true,
          "internalType": "uint256",
          "name": "tokenId",
          "type": "uint256"
        }
      ],
      "name": "Approval",
      "type": "event"
    },
    {
      "anonymous": false,
```

Рис. 2.3.2 Скомпільований контракт

Після компіляції контракту ми готуємо міграцію для деплою контракту в мережу.

Код міграції:

```
const NftMarket = artifacts.require('NftMarket');
```

```
module.exports = function (deployer) {  
  deployer.deploy(NftMarket)  
}
```

Далі за допомогою команди `truffle migrate` деплоємо контракт в мережу.

```
sluzbabezopasnosti@MBP-Sluzba frontend % truffle migrate  
  
Compiling your contracts...  
=====
```

```
> Everything is up to date, there is nothing to compile.
```

```
  
Starting migrations...  
=====
```

```
> Network name:      'development'  
> Network id:        5777  
> Block gas limit:   6721975 (0x6691b7)
```

```
  
1_nftMarket_migration.js  
=====
```

```
Deploying 'NftMarket'
```

```
> transaction hash:  0xd99c755cb5239122b068502392d567bb595b497b3f2c214cfb56b486ef8e93c9  
> Blocks: 0         Seconds: 0  
> contract address: 0xAFF734b775Bf86c1b2f23e830EbE19Ce8EEC8F4f  
> block number:     92  
> block timestamp:  1703772335  
> account:          0x4233B5Fa5F794B57076E2039a43D76537cB97196  
> balance:          99.029582800909779867  
> gas used:         3851734 (0x3ac5d6)  
> gas price:        2.64329377 gwei  
> value sent:       0 ETH  
> total cost:       0.01018126448589718 ETH
```

```
  
> Saving artifacts  
=====
```

```
> Total cost:       0.01018126448589718 ETH
```

```
  
Summary  
=====
```

```
> Total deployments: 1  
> Final cost:       0.01018126448589718 ETH
```

Рис. 2.3.3 Результат успішного деплою контракту

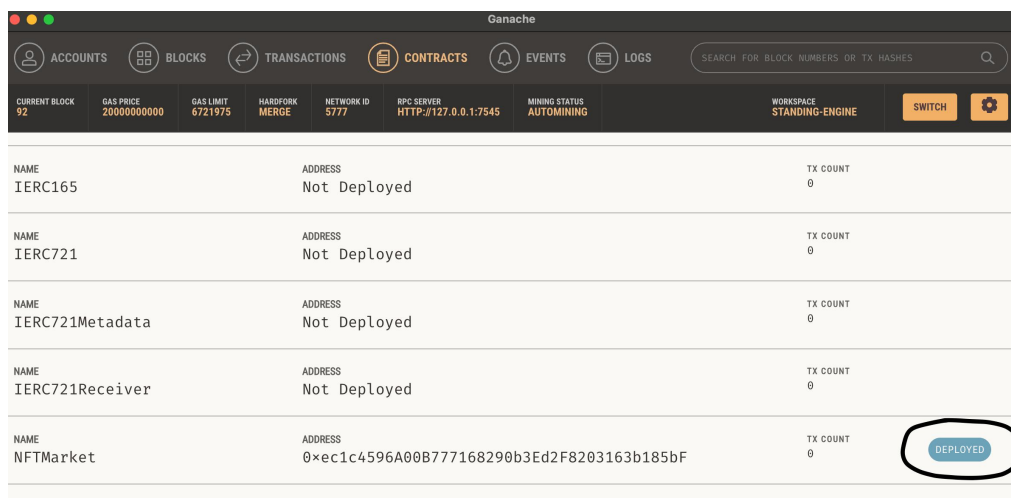


Рис. 2.3.4 Результат успішного деплою контракту

2.4 Розробка веб інтерфейсу для взаємодії з контрактом

2.4.1 Створення дизайну

Для реалізації зручного та зрозумілого інтерфейсу було проведено дослідження останніх тенденцій у сфері UI/UX дизайну та створено власний унікальний дизайн за допомогою застосунку **Figma**.

Figma - це онлайн-редактор, призначений для створення макетів, прототипів і дизайну користувацького інтерфейсу (UI) і користувацького досвіду (UX). Це потужний інструмент, який дає змогу дизайнерам працювати в режимі реального часу, спільно редагувати та ділитися своїми проєктами в хмарі.

Figma використовується веб-дизайнерами, UI/UX-дизайнерами, графічними дизайнерами, продукт-менеджерами, розробниками та іншими фахівцями в галузі цифрового дизайну. Вона стала особливо популярною серед команд, що працюють віддалено, оскільки Figma полегшує спільну роботу та обмін ідеями.

Фігма може використовуватися для створення дизайну мобільних застосунків і адаптивного дизайну, що особливо важливо в даний час, коли більшість користувачів використовують мобільні пристрої для доступу до інтернету.

Figma має безліч переваг для роботи, включно з можливістю роботи в режимі реального часу, спільною роботою, доступністю та зручністю використання. Вона дає змогу створювати компоненти, спрощуючи процес

дизайну та повторного використання елементів інтерфейсу. Крім того, Figma дає змогу створювати прототипи та робити презентації своїх проєктів.

Figma не обмежується роботою тільки з векторною графікою, а й підтримує використання растрових зображень. Це може бути особливо корисним для дизайнерів, які хочуть створити інтерактивні прототипи з використанням реалістичних зображень.

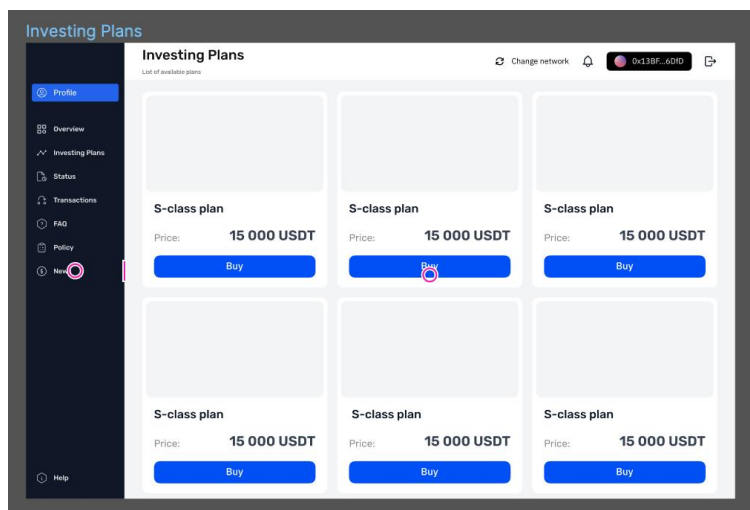


Рис. 2.4.1 Дизайн сторінки з інвестиційними планами

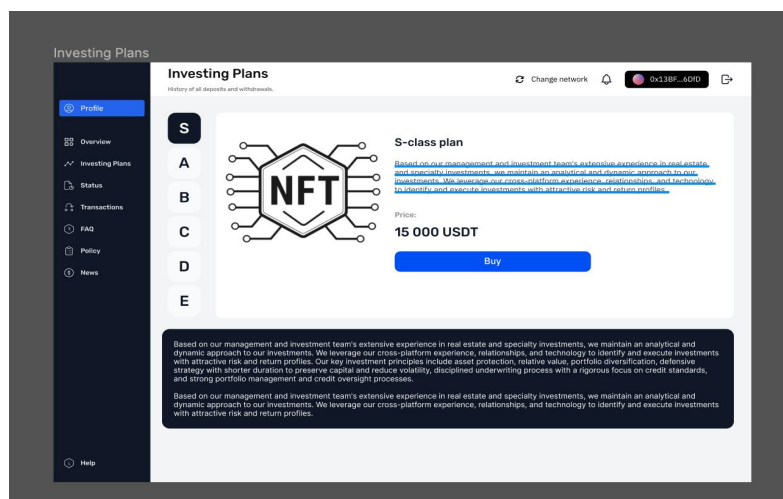


Рис. 2.4.2 Дизайн сторінки з інформацією про план

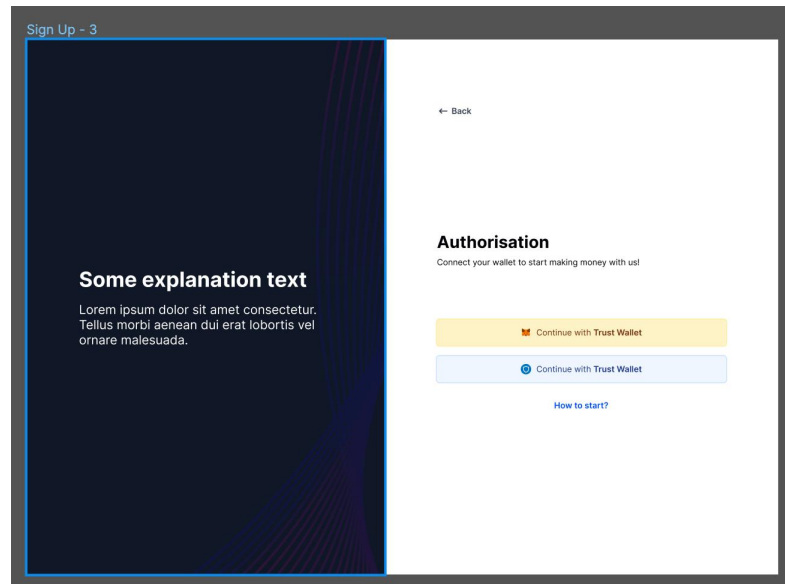


Рис. 2.4.3 Дизайн сторінки логіну

2.4.2 Створення веб додатку

На основі підготовленого дизайну було реалізоване веб рішення на базі технології Next.JS. Були реалізовані наступні сторінки:

1. Профілю користувача
2. Опис основного функціоналу проєкту
3. Сторінка яка відображає інформацію про інвестиційні плани
4. Сторінка де користувач може придбати NFT і прочитати повну інформацію про інвестицію
5. Колекція NFT
6. Сторінка Privacy Policy
7. Login

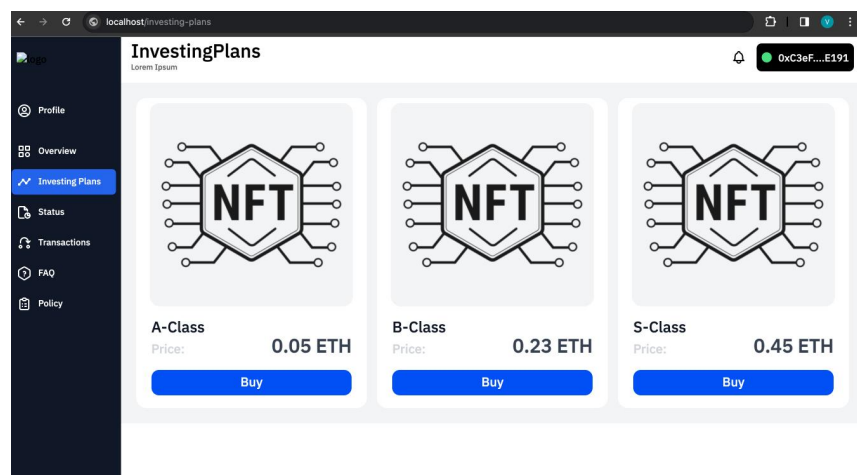


Рис. 2.4.4 Реалізований веб інтерфейс

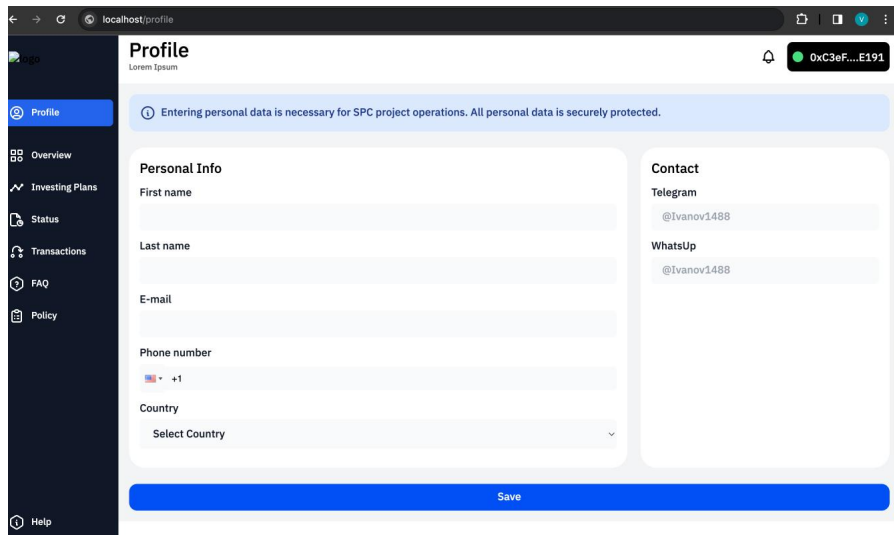


Рис. 2.4.5 Кабінет користувача

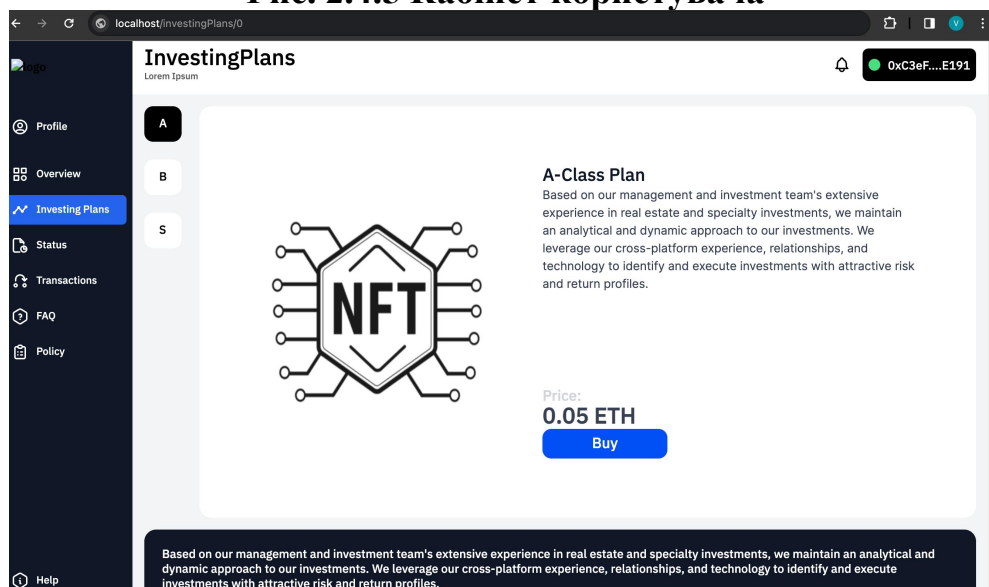


Рис. 2.4.5 Інформація про NFT

Для того щоб наш веб інтерфейс міг взаємодіяти з метамаском та контрактом і також обмежити доступ користувачам без підключеного гаманця були продумані та реалізованні 2 провайдери:

1. Web3Provider(Контролює стан контракту та підключеного гаманця)
2. AccountLocker(Блокує акаунт користувача без підключеного гаманця)

Код Web3Provider:

```
import { createContext, useContext, useEffect, useState } from "react";

import { Web3State, createDefaultState, createWeb3State, loadContract } from
"./utils";

import { providers } from 'ethers';

import { setupHook } from "@hooks";
```

```

import { MetaMaskInpageProvider } from "@metamask/providers";
import { NftMarketContract } from "@types/nftMarketContract";

function pageReload() {
  window.location.reload()
}

const handleAccount = (ethereum: MetaMaskInpageProvider) => async () => {
  const isLocked = !(await ethereum._metamask.isUnlocked());
  if (isLocked) {
    pageReload();
  }
}

const setGlobalListeners = (ethereum: MetaMaskInpageProvider) => {
  ethereum.on("chainChanged", pageReload);
  ethereum.on("accountsChanged", handleAccount(ethereum));
}

const removeGlobalListeners = (ethereum: MetaMaskInpageProvider) => {
  ethereum?.removeListener("chainChanged", pageReload);
  ethereum?.removeListener("accountsChanged", handleAccount)
}

const Web3Context = createContext<Web3State>(createDefaultState());

interface Web3ProviderProps {
  children: React.ReactNode
}

```

```

export default function Web3Provider({ children }: Web3ProviderProps) {
  const [web3Api, setWeb3Api] = useState<Web3State>(createDefaultState());

  useEffect(() => {
    async function initWeb3() {
      try {
        const provider = new providers.Web3Provider(window.ethereum as any);
        const contract = await loadContract("NftMarket", provider);
        console.log(contract)

        const signer = provider.getSigner();
        const signedContract = contract.connect(signer);

        setGlobalListeners(window.ethereum);
        setWeb3Api(createWeb3State({
          ethereum: window.ethereum,
          provider,
          contract: signedContract as unknown as NftMarketContract,
          isLoading: false
        })))
      } catch (e: any) {
        console.log(e)
        setWeb3Api((api) => createWeb3State({
          ...api as any,
          isLoading: false
        })))
      }
    };
    initWeb3();
    return () => removeGlobalListeners(window.ethereum)
  }, []);
}

```

```
}, [])
```

```
return (  
  <Web3Context.Provider value={web3Api}>  
    {children}  
  </Web3Context.Provider>  
)  
}
```

```
export function useHooks() {  
  const { hooks } = useWeb3();  
  return hooks  
}  
  
export function useWeb3() {  
  return useContext(Web3Context)  
}
```

Після того, як успішно були написані та прокинуті провайдери, ми можемо взаємодіяти з користувацьким інтерфесом MetaMask та нашим контрактом. Після підключення гаманця та заповнення профілю користувач може придбати одну або декілька з представлених на маркеті NFT. Після покупки цифровий актив буде зафіксований за користувачем і за активом смарт контракт буде нараховувати проценти. Після закінчення терміну дії інвестиційного плану NFT буде спалена а користувач отримає назад свої кошти та нараховані дивіденти.

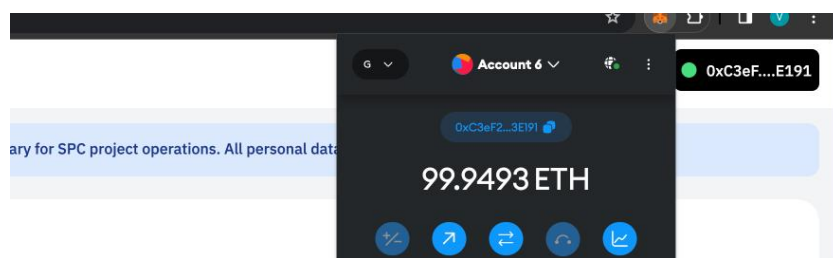


Рис. 2.4.6 Підключний до сайту гаманець

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОЕКТУ, МІНТ NFT, ВІДСТЕЖЕННЯ ПОДІЙ ТА ТРАНЗАКЦІЙ

3.1 Тестування проекту

За допомогою створеного в розділі 2 гаманця MetaMask підключаємось до застосунку і заповнюємо профіль користувача для здійснення транзакцій.

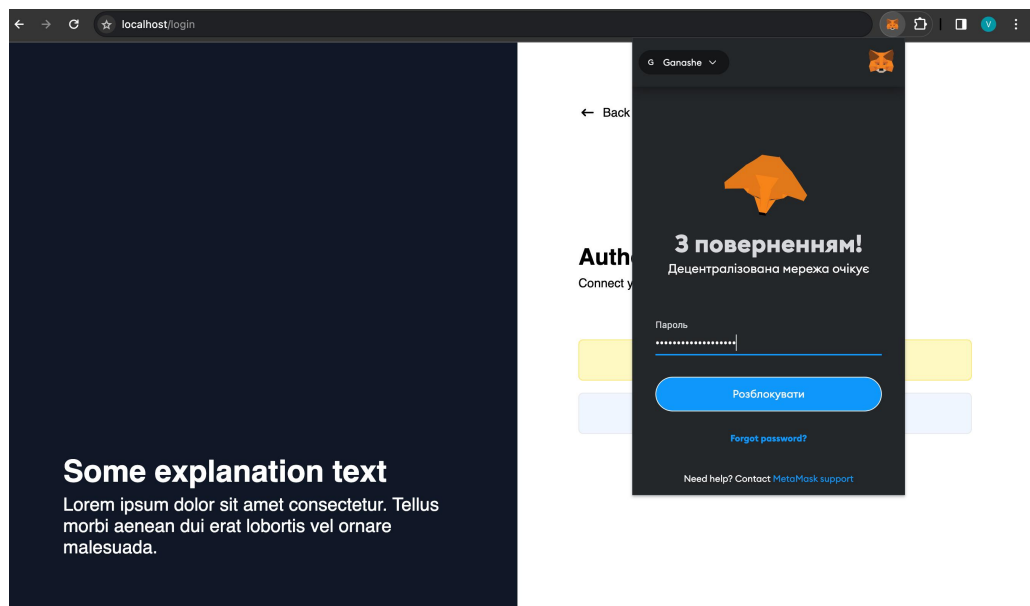


Рис. 3.1 Підключення за допомогою MetaMask

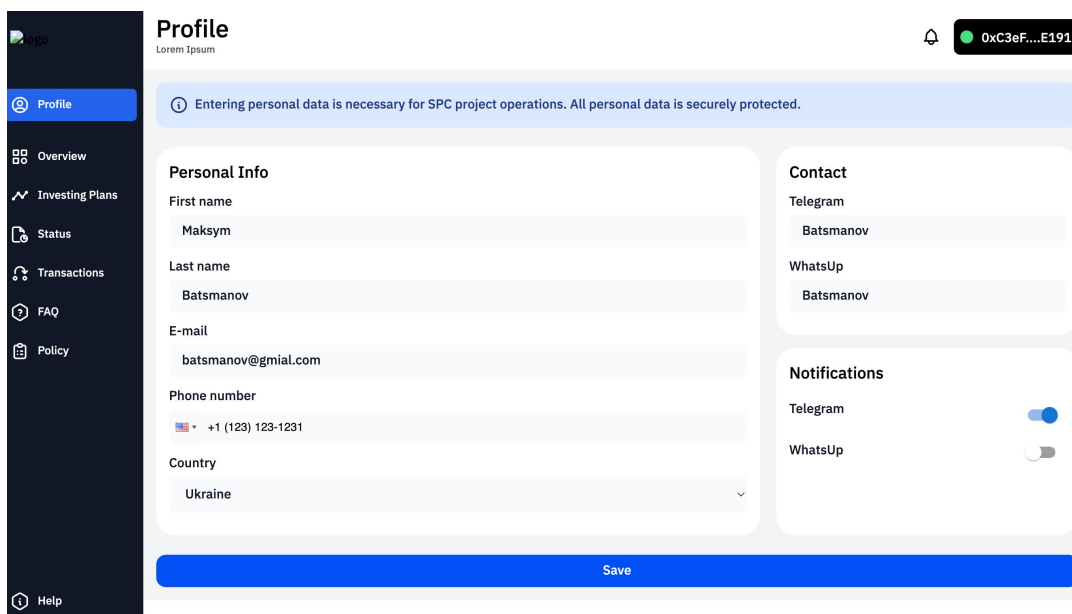


Рис. 3.2 Успішно заповнення профіль користувача

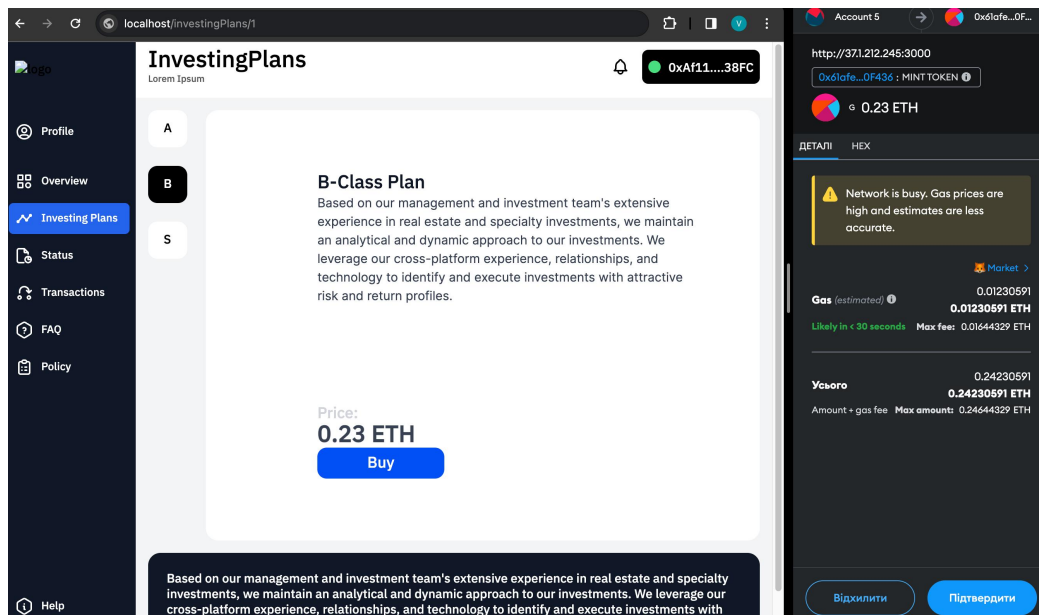


Рис. 3.3 Ініціалізація транзакції

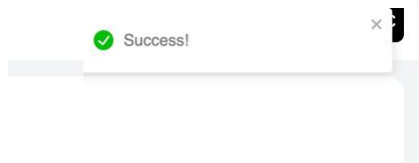


Рис. 3.4 Успішний мінт нового токена

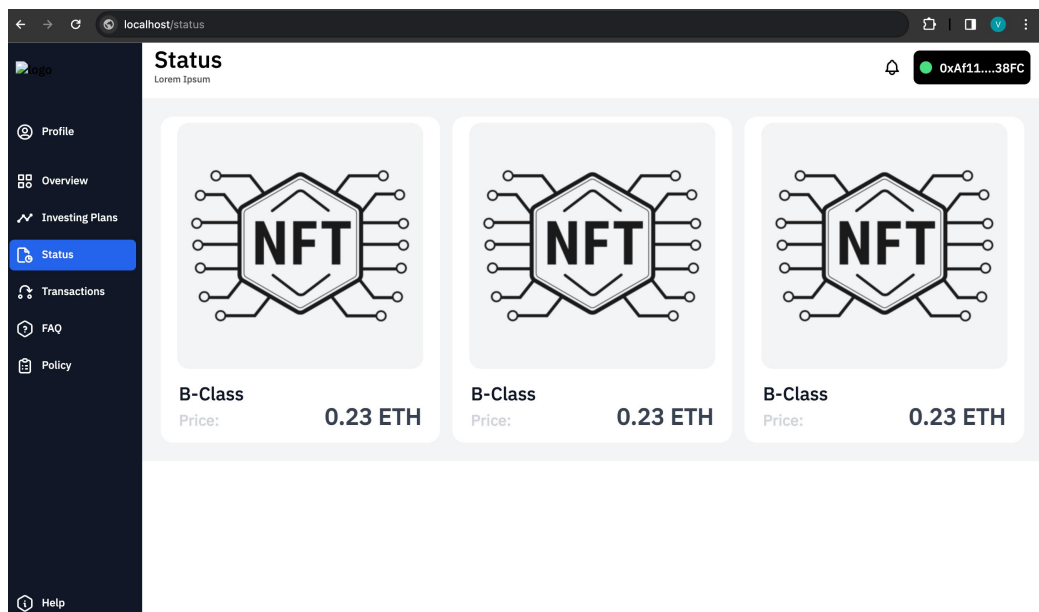


Рис. 3.5 Перевірка наявності NFT у користувача

Після зміни гаманця колекція одразу оновлюється в конкретному наведеному прикладі колекція пуста. Дані про поточну колекцію користувача та нараховані дивіденди зберігається та отримується з блокчейну, що завжди забезпечить швидкий та надійний доступ до активів

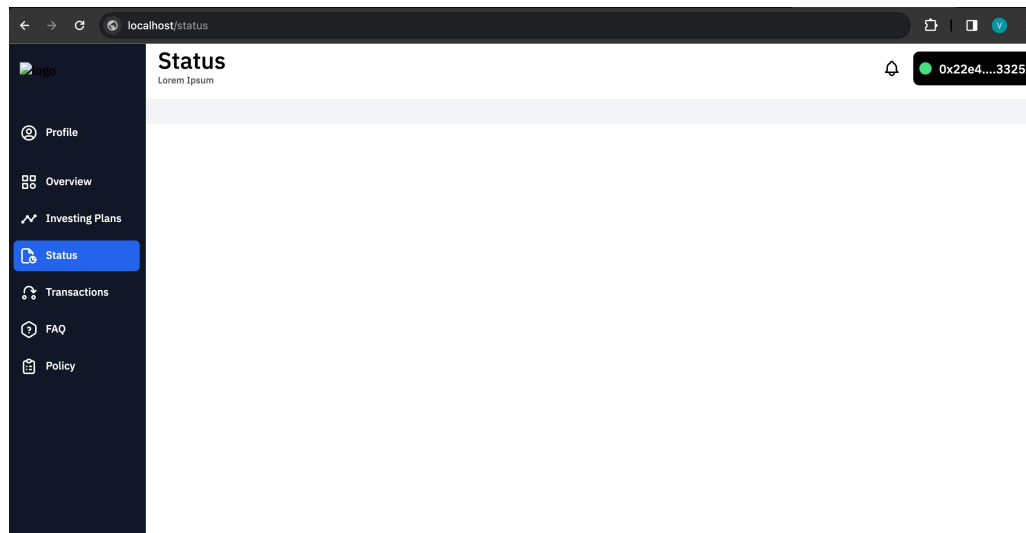


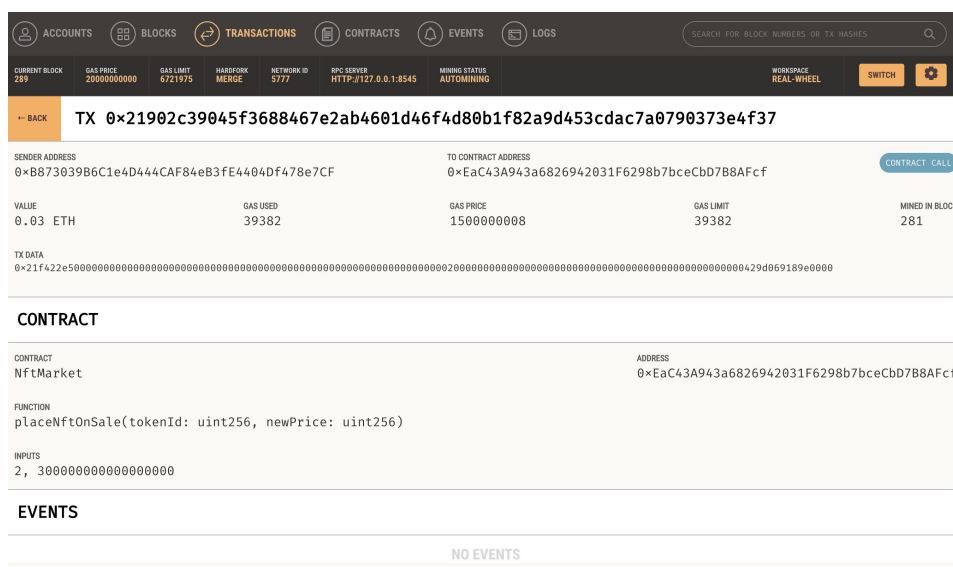
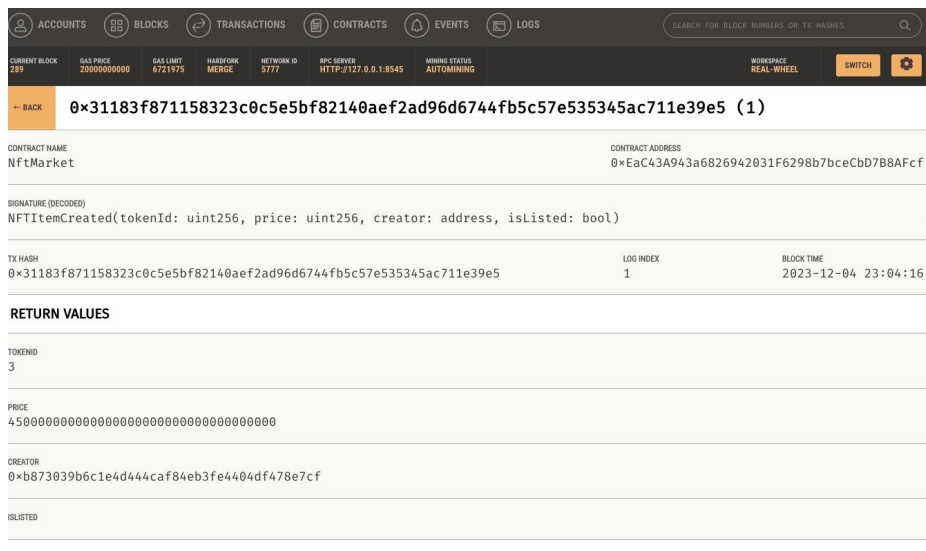
Рис. 3.6 Пуста колекція

3.2 Відстеження подій блокчейну

Функціонал програми Ganache дозволяє нам окремо перегляди івенти та транзакції, В залежності від викликаної функції івенти можуть містити в собі різну інформацію таку як додаткова інформація про токен, ім'я користувача та ін. Також отримуємо загальну інформацію: хеш транзакції(унікальний id), час, назву події або транзакції, статус.

EVENT NAME NFTItemCreated			
CONTRACT NftMarket	TX HASH 0x31183f871158323c0c5e5bf82140aef2ad96d6744fb5c57e535345ac711e39e5	LOG INDEX 1	BLOCK TIME 2023-12-04 23:04:16
EVENT NAME Transfer			
CONTRACT NftMarket	TX HASH 0x31183f871158323c0c5e5bf82140aef2ad96d6744fb5c57e535345ac711e39e5	LOG INDEX 0	BLOCK TIME 2023-12-04 23:04:16
EVENT NAME Transfer			
CONTRACT NftMarket	TX HASH 0x3a5a61c615c1f9c062f1ba375bcc3e6ced73cb823feedc6f44fd50acaf3158af	LOG INDEX 1	BLOCK TIME 2023-12-04 22:34:03
EVENT NAME Approval			
CONTRACT NftMarket	TX HASH 0x3a5a61c615c1f9c062f1ba375bcc3e6ced73cb823feedc6f44fd50acaf3158af	LOG INDEX 0	BLOCK TIME 2023-12-04 22:34:03

Рис. 3.7 Відстеження подій в блокчейні



В останньому розділі протестовано основний функціонал веб застосунку, перевірено коректність роботи смарт-контракту та локальної, перевірено контракт на відповідність всім стандартам.

Проведено оцінку ефективності досліджуваних технологій, розроблено стратегію логування подій блокчейна, для збереження цілісності даних та правильного їх надання користувачам.

ВИСНОВКИ

Технологія блокчейн вносить кардинальні зміни у підходи до взаємодії користувачів та сторонніх учасників. Тепер головне питання — не які елементи системи є найнадійнішими, а як правильно використовувати блокчейн у сучасних реаліях. Основна мета цього дослідження — адаптація блокчейн-технологій до сучасних вимог. Реалізація ідеальної моделі блокчейну є складною задачею, але можна вибрати гнучкий підхід, який передбачає:

- Визначення галузей та ситуацій, де блокчейн не є оптимальним рішенням через його обмеження.
- Чітке формулювання вимог до таких систем від користувачів.
- Створення адаптивних моделей блокчейну, що задовольняють потреби користувачів в різних умовах.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. **Ethereum Project** [Електронний ресурс]. Режим доступу: <https://www.ethereum.org/>
2. **Bitcoin and Cryptocurrencies: Law Enforcement Investigative Guide** [Електронний ресурс]. Режим доступу: <http://www.iacpsybercenter.org/wpcontent/uploads/2018/03/Bitcoin.pdf>
3. **Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications / Andrew Hoffman., 2020. – 330 с.**
4. **CryptoNote** [Електронний ресурс]. Режим доступу: <https://cryptonote.org/>
5. **NFTs Are the Biggest Internet Craze. Do They Work for Sneakers?** Режим доступу: <https://www.wsj.com/articles/nfts-and-fashion-collectors-pay-big-money-forvirtual-sneakers-11615829266>
6. **‘You can now buy a Tesla with bitcoin,’ CEO Elon Musk said. But it could mean a big tax bill.** Режим доступу: <https://www.cnbc.com/2021/03/24/buying-a-teslausing-bitcoin-it-could-mean-a-tax-bill.html>
7. **Why Mastercard is bringing crypto onto its network.** Режим доступу: <https://www.mastercard.com/news/perspectives/2021/why-mastercard-isbringing-crypto-onto-our-network/>
8. **NFTs Govern Ownership of the Token, Not the Underlying IP Rights.** Режим доступу: <https://www.thefashionlaw.com/nfts-what-are-you-buying-and-what-do-youactually-own/>
9. **Atomichub.** Режим доступу: <https://atomichub.io/>
10. **ERC-721 NON-FUNGIBLE TOKEN STANDARD.** Режим доступу: <https://ethereum.org/en/developers/docs/standards/tokens/erc-721/>
11. **INTRODUCTION TO SMART CONTRACTS.** Режим доступу: <https://ethereum.org/en/developers/docs/smart-contracts>
12. **OpenSea.** Режим доступу: <https://opensea.io/>
13. **Crypto.** Режим доступу: <https://crypto.com/nft>

- 14. The Math Behind the Bitcoin Protocol. Режим доступа:**
<https://www.coindesk.com/math-behind-bitcoin>
- 15. Solidity. Режим доступа:** **<https://docs.soliditylang.org/>**
- 16. Truffle suite. Режим доступа:** **<https://www.trufflesuite.com/>**
- 17. Nodejs. Режим доступа:** **<https://nodejs.org/>**
- 18. Metamask. Режим доступа:** **<https://metamask.io/>**
- 19. Pinata. URL:** **<https://pinata.cloud>**

ДОДАТКИ

package.json:

```

{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "dev:blockchain": "truffle development",
    "lint": "next lint",
    "genContractType": "abi-types-generator './public/contracts/NftMarket.json' --
output='./types' --name=nftMarketContract --provider=ethers_v5"
  },
  "dependencies": {
    "@apollo/client": "^3.8.4",
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@metamask/providers": "^14.0.2",
    "@mui/material": "^5.14.10",
    "@openseppelin/contracts": "^4.5.0",
    "@tabler/icons-react": "^2.35.0",
    "@truffle/hdwallet-provider": "^2.1.15",
    "@types/node": "20.5.1",
    "@types/react": "18.2.20",
    "@types/react-dom": "18.2.7",
    "autoprefixer": "10.4.15",
    "axios": "^1.3.6",

```

```

"deepmerge": "^4.3.1",
"ethereum-abi-types-generator": "^1.3.4",
"ethers": "5.6.2",
"framer-motion": "^10.16.4",
"lodash": "^4.17.21",
"next": "13.4.19",
"next-auth": "^4.22.3",
"postcss": "8.4.28",
"react": "18.2.0",
"react-country-region-selector": "^3.6.1",
"react-dom": "18.2.0",
"react-icons": "^4.10.1",
"react-phone-input-2": "^2.15.1",
"react-toastify": "^9.1.3",
"solc": "^0.8.20",
"swr": "^2.2.4",
"tailwindcss": "3.3.3",
"truffle": "^5.11.5",
"typescript": "5.1.6",
"wagmi": "^0.6.7"
},
"devDependencies": {
"@types/lodash": "^4.14.199",
"@types/node": "^20",
"@types/react": "^18",
"@types/react-dom": "^18",
"autoprefixer": "^10.4.16",
"eslint": "^8",
"eslint-config-next": "14.0.3",
"postcss": "^8.4.31",

```

```
"tailwindcss": "^3.3.5",
"typescript": "^5"
}
}
```

truffle-config.js:

```
const HDWalletProvider = require("@truffle/hdwallet-provider");
const mnemonic = "";

module.exports = {
  contracts_build_directory: "./public/contracts",
  networks: {
    development: {
      host: "127.0.0.1", // Localhost (default: none)
      port: 7545, // Standard Ethereum port (default: none)
      network_id: "*", // Any network (default: none)
    },
    // sepolia: {
    //   provider: function () {
    //     return new HDWalletProvider(mnemonic, 'https://wild-maximum-
wind.ethereum-sepolia.quiknode.pro/12497e6874c137f4eb46dc28f7e574c54f51925a')
    //   },
    //   network_id: 4868603,
    //   gas: 4000000, // Adjust the gas limit as per your requirements
    //   gasPrice: 10000000000, // Set the gas price to an appropriate value
    //   confirmations: 2, // Set the number of confirmations needed for a transaction
    //   timeoutBlocks: 200, // Set the timeout for transactions
    //   skipDryRun: true // Skip the dry run option
    // }
  },
}
```

```
compilers: {  
  solc: {  
    version: "0.8.13",  
  }  
}  
};
```

tsconfig.json:

```
{  
  "compilerOptions": {  
    "baseUrl": ".",  
    "target": "es5",  
    "lib": ["dom", "dom.iterable", "esnext"],  
    "allowJs": true,  
    "skipLibCheck": true,  
    "strict": true,  
    "noEmit": true,  
    "esModuleInterop": true,  
    "module": "esnext",  
    "moduleResolution": "bundler",  
    "resolveJsonModule": true,  
    "isolatedModules": true,  
    "jsx": "preserve",  
    "incremental": true,  
    "paths": {  
      "@/*": ["/.*"],  
      "@ui": ["components/UI"],  
      "@ui/*": ["components/UI/*"],  
      "@providers/*": ["components/Providers/*"],  
      "@providers": ["components/Providers"],  
    }  
  }  
}
```

```

"@hooks/*": ["components/Hooks/*"],
"@hooks": ["components/Hooks"],
"@_types/*": ["types/*"]
}
},
"include": [
"process.d.ts",
"next-env.d.ts",
"next-auth.d.ts",
"**/*.ts",
"**/*.tsx",
"next-i18next.config.js",
"lib/sse/sse.js"
],
"exclude": ["node_modules"],
"paths": {
"@/*": [
"./*"
]
}
}

```

types/hook.ts:

```

import { MetaMaskInpageProvider } from "@metamask/providers";
import { providers } from "ethers";
import { SWRResponse } from "swr";
import { NftMarketContract } from "../nftMarketContract";

export type Web3Deps = {
  provider: providers.Web3Provider;

```

```
contract: NftMarketContract;  
ethereum: MetaMaskInpageProvider;  
isLoading: boolean;  
}
```

```
export type CryptoSWRResponse<D = any, R = any> = SWRResponse<D> & R;
```

```
export type CryptoHandlerHook<D = any, R = any, P = any> = (params?: P) =>  
CryptoSWRResponse<D, R>;
```

```
export type CryptoHookFactory<D = any, R = any, P = any> = {  
(deps: Partial<Web3Deps>): CryptoHandlerHook<D, R, P>  
};
```

types/nft.ts:

```
export type Trait = 'attack' | "health" | "speed";
```

```
export type NftAttribute = {  
  trait_type: Trait;  
  value: string;  
};
```

```
export type NftType = {  
  name: string;  
  description: string;  
  image: string;  
  attributes: NftAttribute[];  
}
```

```
export type NftMeta = {
```

```
name: string;
description: string;
image: string;
attributes: NftAttribute[];
};
```

```
export type NftCore = {
tokenId: number;
price: number;
creator: string;
isListed: boolean;
};
```

```
export type Nft = {
meta: NftMeta
} & NftCore;
```

pages/login.tsx:

```
import React, { useEffect, useState } from "react";
import { IconArrowNarrowLeft } from '@tabler/icons-react';
import Link from "next/link";
import { useAccount } from "@components/Hooks";
import { useRouter } from "next/router";
import { toast } from "react-toastify";
```

```
export default function Login() {
const { account } = useAccount();
const router = useRouter();
```

```
if (account.data) {
router.push('/profile')
```

```

    }

    return (
      <div className="w-full flex h-screen font-istok-web">
        <div className="w-1/2 p-20 bg-[#111827] flex flex-col justify-between items-
start">
          <img
            src='/images/WhiteHeader.svg'
            className="w-68 h-9"
          />
          <div>
            <p className="font-bold text-[40px] leading-12 mt-4">Some explanation
text</p>
            <p className="font-normal text-2xl leading-7">Lorem ipsum dolor sit amet
consectetur. Tellus morbi aenean dui erat lobortis vel ornare malesuada.</p>
          </div>

        </div>

        <div className="w-1/2 p-20 flex flex-col text-black">
          <Link
            className="flex gap-2 mb-40"
            href="/"
          >
            <IconArrowNarrowLeft />
            Back
          </Link>
          <p className="font-bold text-3xl leading-9 mb-1.5">Authorisation</p>
          <p className="font-normal text-base leading-5">Connect your wallet to start
making money with us!</p>
          <button

```

```
className="flex rounded-md w-full justify-center gap-1.5 font-normal text-base  
leading-5 text-yellow-900 bg-yellow-100 border border-inherit border-yellow-300  
border-opacity-70 py-4 mt-16 mb-4"
```

```
onClick={(e) => {  
  e.preventDefault()  
  if (!account.data && account.isInstalled) {  
    account.connect();  
    router.push('/profile');  
  } else if (account.data) {  
    router.push('/profile')  
  } else {  
    toast.error('Wallet error')  
  }  
}}
```

```
>  
<img  
src='/images/metamask.svg'  
className="h-5 w-5"  
/>
```

```
Continue with <span className='font-semibold'>Meta Mask</span>
```

```
</button>
```

```
<button
```

```
className="flex rounded-md w-full justify-center gap-1.5 font-normal text-base  
leading-5 text-blue-900 bg-blue-50 border border-inherit border-blue-100 border-  
opacity-70 py-4 mb-8"
```

```
onClick={(e) => {  
  e.preventDefault()  
  if (!account.data && account.isInstalled) {  
    account.connect();  
    router.push('/profile');
```

```

    } else if (account.data) {
      router.push('/profile')
    } else {
      toast.error('Wallet error')
    }
  }}
}
<img
  src='/images/trustWallet.svg'
  className="h-5 w-5"
/>
Continue with <span className='font-semibold'>Trust Wallet</span>
</button>
<Link
  href='#'
  className="text-[#0050F6] font-semibold text-base leading-5 mx-auto"
>
How to start?
</Link>
</div>
</div>
)
}

```

frontend/pages/profile.tsx:

```

import React, { ChangeEvent, useEffect, useState } from "react"
import AppLayout from "../components/UI/profile/layout/appLayout"
import { IconInfoCircle } from '@tabler/icons-react';
import { useMutation, useQuery } from "@apollo/client";
import { USER_QUERY } from "../graphql/queries/user";

```

```

import { UPDATE_USER_MUTATION,
UPDATE_USER_NOTIF_CONFIG_MUTATION } from "../graphql/mutations/user";
import { useSession } from 'next-auth/react';
import { toast } from "react-toastify";
import { Backdrop, CircularProgress, Switch } from "@mui/material";
import PhoneInput from 'react-phone-input-2'
import { CountryDropdown } from 'react-country-region-selector';
import 'react-phone-input-2/lib/style.css'
import ValidateInput from "../components/validationInput";
import { useAccount } from "@components/Hooks";
export default function Profile() {

const { account } = useAccount();

const [name, setName] = useState<string>("");
const [nameError, setNameError] = useState<string>();

const [lastName, setLastName] = useState<string>("");
const [lastNameError, setLastNameError] = useState<string>();

const [email, setEmail] = useState<string>();
const [emailError, setEmailError] = useState("")

const [phone, setPhone] = useState<string>();
const [country, setCountry] = useState<string>();

const [telegram, setTelegram] = useState<string>();
const [telegramNotification, setTelegramNotification] = useState<boolean>();

const [whatsUp, setWhatsUp] = useState<string>();

```

```

const [whatsUpNotification, setWhatsUpNotification] = useState<boolean>();

function onNameChange(e: ChangeEvent<HTMLInputElement>) {
  if (e.currentTarget.value.length < 2) {
    setNameError('Name must be at least 2 characters')
  } else {
    setNameError("")
  }
  setName(e.currentTarget.value)
}

function onLastNameChange(e: ChangeEvent<HTMLInputElement>) {
  if (e.currentTarget.value.length < 2) {
    setLastNameError('LastName must be at least 2 characters')
  } else {
    setLastNameError("")
  }
  setLastName(e.currentTarget.value)
}

function onMailChange(e: ChangeEvent<HTMLInputElement>) {
  const re = /^[^<>()[]\.\,\;\s@""]+(\.[^<>()[]\.\,\;\s@""]+)*|(\."+")@((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\]|((\[a-zA-Z\0-9]+\.)+[a-zA-Z]{2,})))$/;
  if (!re.test(String(e.currentTarget.value).toLowerCase())) {
    setEmailError('Input valid mail')
  }
  if (!e.currentTarget.value) {
    setEmailError('Input valid mail');
  }
  } else setEmailError("");
  setEmail(e.currentTarget.value);
}

```

```

const label = { inputProps: { 'aria-label': 'Switch demo' } };

return (
  <AppLayout>
    <Backdrop
      sx={{ color: '#fff', zIndex: (theme) => theme.zIndex.drawer + 1 }}
      open={account.isLoading}
    >
      <CircularProgress color="inherit" />
    </Backdrop>
    <div className="bg-blue-100 text-blue-900 flex gap-2 items-center rounded-xl
p-4">
      <IconInfoCircle />
      <p>Entering personal data is necessary for SPC project operations. All personal
data is securely protected.</p>
    </div>

    <div className="flex gap-5 my-6">
      <div className="w-2/3 bg-white p-4 rounded-3xl text-gray-900">
        <h1 className="font-medium text-xl leading-8 mb-2 text-black">Personal
Info</h1>
        <p className="leadng-6 font-normal mb-1">First name</p>
        <ValidateInput
          className="w-full rounded-md p-2 px-4 mb-3"
          error={nameError}
          value={name}
          onChange={onNameChange}
          type="text"
        />
        <p className="leadng-6 font-normal mb-1">Last name</p>

```

```

<ValidateInput
className="w-full rounded-md p-2 px-4 mb-3"
error={lastNameError}
value={lastName}
onChange={onLastNameChange}
type="text"
/>

<p className="leading-6 font-normal mb-1">E-mail</p>
<ValidateInput
className="w-full rounded-md p-2 px-4 mb-3"
error={emailError}
value={email}
onChange={onMailChange}
type="text"
/>

<p className="leading-6 font-normal mb-1">Phone number</p>
<PhoneInput
country={'us'}
value={phone}
onChange={phone => setPhone(phone)}
containerClass='!rounded-2xl !border-none !py-1'
buttonClass='!rounded-md !bg-gray-50 !border-none' // Apply rounded-lg class to
change container's border-radius
inputClass='!w-full !rounded-md !text-black !border-none !py-2 !bg-gray-50'
/>

<p className="leading-6 font-normal mb-1 mt-3">Country</p>
<CountryDropdown
classes="w-full bg-gray-50 rounded-md p-2 px-4 mb-3 border-none
focus:outline-none"
value={country || ""}

```

```

    onChange={(val) => setCountry(val)}
  />
</div>
<div className="w-1/3 text-gray-900 flex flex-col gap-4">
  <div className={` ${ (telegram || whatsUp) ? 'h-[50%]' : 'h-full'} w-full bg-white
p-4 rounded-3xl`} >
    <h1 className="font-medium text-xl leading-8 mb-2 text-black">Contact</h1>
    <p className="leading-6 font-normal mb-1">Telegram</p>
    <input
      className="w-full bg-gray-50 rounded-md p-2 px-4 mb-3"
      placeholder="@Ivanov1488"
      value={telegram}
      onChange={(e) => setTelegram(e.target.value)}
    />
    <p className="leading-6 font-normal mb-1">WhatsUp</p>
    <input
      className="w-full bg-gray-50 rounded-md p-2 px-4 mb-3"
      placeholder="@Ivanov1488"
      value={whatsUp}
      onChange={(e) => setWhatsUp(e.target.value)}
    />
  </div>
  {(telegram || whatsUp) && <div className={`w-full h-[50%] bg-white p-4
rounded-3xl`} >
    <h1 className="font-medium text-xl leading-8 mb-2 text-black items-
center">Notifications</h1>
    {
      telegram && <div className="w-full flex justify-between my-2 mt-4">
        <p className="leading-6 font-normal mb-1">Telegram</p>

```

```

        <Switch    {...label}    checked={telegramNotification}    onChange={()    =>
setTelegramNotification(!telegramNotification)} />
    </div>
    }
    {
    whatsUp && <div className="w-full flex justify-between items-center">
    <p className="leadng-6 font-normal mb-1">WhatsUp</p>
    <Switch    {...label}    checked={whatsUpNotification}    onChange={()    =>
setWhatsUpNotification(!whatsUpNotification)} />
    </div>
    }

    </div>}
    </div>

    </div>

    <button
    className="text-white bg-[#0050F6] text-center rounded-xl w-full py-2"
    onClick={updateProfile}
    >
    Save
    </button>
    </AppLayout>
    )
    }

```

pages/investingPlans/index.tsx:

```

import React from "react"
import AppLayout from "../../components/UI/profile/layout/appLayout"

```

```

import Link from "next/link"
import nftsData from '../data/nfts.json'

export default function InvestingPlans() {

  return (
    <AppLayout>
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3">
        {
          nftsData?.map((nft, index) =>
            <div className="bg-white rounded-2xl flex flex-col m-2">
              <div className="bg-gray-100 rounded-2xl max-w-[20rem] mx-auto mt-2">
                <img src={nft.image} className="rounded-2xl" />
              </div>
              <div className="mx-6 mt-5">
                <p className="text-gray-900 font-medium text-2xl leading-7">{nft.name}</p>
                <div className="flex justify-between items-end">
                  <p className="font-normal text-gray-300 text-xl leading-6">Price:</p>
                  <p className="font-semibold text-gray-700 leading-8 text-3xl">{nft.price}
ETH</p>
                </div>
              </div>
              <div>
                <Link
                  href={` /investingPlans/${index} `}
                  className="py-1.5 text-white text-center bg-[#0050F6] rounded-xl my-5 mx-6
text-xl"
                >
                  Buy
                </Link>
              </div>
            </div>
          )
        }
      </div>
    </AppLayout>
  )
}

```

```

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

```

pages/investingPlans/[id].tsx:

```

import React from "react"
import AppLayout from "../../components/UI/profile/layout/appLayout"
import Link from "next/link"
import nftsData from "../../data/nfts.json"
import { Router, useRouter } from "next/router"
import { BigNumber, ethers } from "ethers"
import { useWeb3 } from "@providers"
import { toast } from "react-toastify"

export default function Plan() {
  const { contract } = useWeb3();
  const router = useRouter()
  const { id } = router.query;
  const numId = parseInt(id as string)
  // console.log(router)
  // if (!id || typeof numId === 'undefined') {
  //   router?.push('/investingPlans')
  //   // return router.push('/investingPlans')
  // }
  const nft = nftsData.find((item, index) => index === numId);

  if (!nft) return

```

```

const mintToken = async () => {
  console.log(contract)
  try {
    console.log(ethers.utils.parseEther(nft.price))
    const trans = await contract?.mintToken(nft.data, ethers.utils.parseEther(nft.price),
  {
    value: ethers.utils.parseEther(nft.price)
  });
    await trans?.wait()
    toast.success("Success!")
  } catch (e: any) {
    toast.error(e.message)
  }
}

return (
  <AppLayout>
    <div className="flex gap-6 ">
      <div className="flex flex-col gap-6">
        {
          nftsData.map((nft, index) => <Link href={`/${investingPlans}/${index}`}
className={`/${index} === numId ? "bg-black text-white" : "bg-white text-black"}
rounded-xl p-3 px-5`>{nft.name.slice(0, 1)}</Link>)
        }
      </div>
      <div className="bg-white rounded-2xl p-20 flex w-full">
        <img src={nft.image} className="max-w-[20rem] rounded-xl" />
        <div className="flex flex-col justify-between ml-16">
          <p className="text-gray-900 font-medium text-2xl leading-7">Stagepoint
{nft.name} Plan</p>

```

```

    <p className="font-inter text-base text-gray-700">{nft.description}</p>
    <div className="flex-col items-end mt-32">
      <p className="font-normal text-gray-300 text-xl leading-6">Price:</p>
      <p className="font-semibold text-gray-700 leading-8 text-3xl">{nft.price}
    ETH</p>
    </div>
    <button
      onClick={mintToken}
      className="py-1.5 text-white text-center text-xl font-medium bg-[#0050F6]
rounded-xl w-1/3"
    >
      Buy
    </button>
  </div>
</div>

<div className="bg-gray-900 text-white p-6 rounded-2xl font-normal text-base
leading-5 space-y-4 mt-4">
  {nft.description}
</div>
</AppLayout>
)
}

```

pages/_app.tsx:

```

import '@styles/globals.css'
import type { AppProps } from 'next/app'
import { ToastContainer } from "react-toastify"
import 'react-toastify/dist/ReactToastify.css';
import { Web3Provider } from '@components/Providers'

```

```

import AccountLocker from '@components/Providers/account'

export default function App({ Component, pageProps }: AppProps<{}>) {

  return (
    <Web3Provider>
      <AccountLocker>
        <Component {...pageProps} />
      </AccountLocker>
      <ToastContainer
        position="top-right"
        autoClose={3000}
        style={{ overflow: "hidden" }}
        hideProgressBar={false}
        newestOnTop={false}
        draggable={false}
        closeOnClick
        pauseOnHover
      />
    </Web3Provider >
  )
}

```

migrations/1_nftMarket_migration.js:

```

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

module.exports = function (deployer) {
  deployer.deploy(NftMarket)
}

```

data/nfts.json:

```
[
  {
    "description": "Based on our management and investment team's extensive
experience in real estate and specialty investments, we maintain an analytical and
dynamic approach to our investments. We leverage our cross-platform experience,
relationships, and technology to identify and execute investments with attractive risk
and return profiles.",
    "image": "https://jade-cheap-toad-
19.mypinata.cloud/ipfs/QmNPap2k1zDLykuXQedsLqfg6e7VPAbQWb7MRkTsg3Pt8r
",
    "name": "A-Class",
    "price": "0.05",
    "data": "https://jade-cheap-toad-
19.mypinata.cloud/ipfs/QmVaUtHnP4T9uBUSi3r395KfcogTWkUcgRrKJPwN7zx2TS
"
  },
  {
    "description": "Based on our management and investment team's extensive
experience in real estate and specialty investments, we maintain an analytical and
dynamic approach to our investments. We leverage our cross-platform experience,
relationships, and technology to identify and execute investments with attractive risk
and return profiles.",
    "image": "https://jade-cheap-toad-
19.mypinata.cloud/ipfs/QmQ8gHa66GWUdRBWvLffjZY46WE42SHoRA7VYkYyqv
UhE4",
    "name": "B-Class",
    "price": "0.23",
```

```

    "data": "https://jade-cheap-toad-19.mypinata.cloud/ipfs/QmR5Rj16NVakCbh23nQhfzCKW6MAxFjkvrGrixU5H39MrQ"
  },
  {
    "description": "Based on our management and investment team's extensive experience in real estate and specialty investments, we maintain an analytical and dynamic approach to our investments. We leverage our cross-platform experience, relationships, and technology to identify and execute investments with attractive risk and return profiles.",
    "image": "https://jade-cheap-toad-19.mypinata.cloud/ipfs/QmUttzLAN4f8GmZptbiAzThUbGVzMKBSSTfW8wTs97hH2n",
    "name": "S-Class",
    "price": "0.45",
    "data": "https://jade-cheap-toad-19.mypinata.cloud/ipfs/QmQYjpHPFy1Aidk1bnt78rTEB8oCSXCkRg75P85iBwb9"
  }
]

```

contracts/NftMarket.sol:

```

// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

import
"@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract NftMarket is ERC721URIStorage, Ownable {

```

```

using Counters for Counters.Counter;

Counters.Counter private _listedItems;
Counters.Counter private _tokenIds;

// All Token Ids in array
uint256[] private _allNfts;

mapping(string => bool) private _usedTokenURIs;
mapping(uint => NFTItem) private _idToNftItem;

mapping(address => mapping(uint => uint)) private _ownedTokens;
mapping(uint => uint) private _idToOwnedIndex;

mapping(uint => uint) private _idToNftIndex;

struct NFTItem {
    uint tokenId;
    uint price;
    address creator;
    bool isListed;
}

event NFTItemCreated (
    uint tokenId,
    uint price,
    address creator,
    bool isListed
);

```

```
uint public listingPrice = 0.025 ether;
```

```
constructor() ERC721("CreaturesNFT", "CNFT") {}
```

```
function setListingPrice(uint newPrice) external onlyOwner {  
    require(newPrice > 0, "Price must be at least 1 wei");  
    listingPrice = newPrice;  
}
```

```
function getNftItem(uint tokenId) public view returns (NFTItem memory) {  
    return _idToNftItem[tokenId];  
}
```

```
function listedItemsCount() public view returns (uint) {  
    return _listedItems.current();  
}
```

```
function tokenURIExists(string memory tokenURI) public view returns (bool) {  
    return _usedTokenURIs[tokenURI] == true;  
}
```

```
function totalSupply() public view returns (uint) {  
    return _allNfts.length;  
}
```

```
function tokenByIndex(uint index) public view returns (uint) {  
    require(index < totalSupply(), "Index out of bounds");  
    return _allNfts[index];  
}
```

```

function tokenOfOwnerByIndex(address owner, uint index) public view returns
(uint) {
    require(index < ERC721.balanceOf(owner), "Index out of bounds");
    return _ownedTokens[owner][index];
}

```

```

function getOwnedNfts() public view returns (NFTItem[] memory) {
    uint ownedItemsCount = ERC721.balanceOf(msg.sender);
    NFTItem[] memory items = new NFTItem[](ownedItemsCount);

```

```

    for (uint i = 0; i < ownedItemsCount; i++) {
        uint tokenId = tokenOfOwnerByIndex(msg.sender, i);
        NFTItem storage item = _idToNftItem[tokenId];
        items[i] = item;
    }

```

```

    return items;
}

```

```

function burnToken(uint tokenId) public {
    _burn(tokenId);
}

```

```

function mintToken(string memory tokenURI, uint price) public payable returns
(uint) {
    // require(!tokenURIExists(tokenURI), "Token URI already exists");
    // require(msg.value == listingPrice, "Price must be qual to listing price");
    _tokenIds.increment();
    _listedItems.increment();
}

```

```

uint newTokenId = _tokenIds.current();
_safeMint(msg.sender, newTokenId);
_setTokenURI(newTokenId, tokenURI);
_createNftItem(newTokenId, price);
_usedTokenURIs[tokenURI] = true;

return newTokenId;
}

function _createNftItem(uint tokenId, uint price) private {
require(price > 0, "Price must be at least 1wei");

_idToNftItem[tokenId] = NFTItem(
tokenId,
price,
msg.sender,
true
);

emit NFTItemCreated(tokenId, price, msg.sender, true);
}

function _beforeTokenTransfer(address from, address to, uint tokenId) internal
virtual override {
super._beforeTokenTransfer(from, to, tokenId);

if (from == address(0)) {
_addTokenToAllTokensEnumeration(tokenId);
} else if (from != to) {
_removeTokenFromOwnerEnumeration(from, tokenId);
}
}

```

```

    }

    if (to == address(0)) {
        _removeTokenFromAllTokensEnumeration((tokenId));
    } else if (to != from) {
        _addTokenToOwnerEnumeration(to, tokenId);
    }
}

function _addTokenToAllTokensEnumeration(uint tokenId) private {
    _idToNftIndex[tokenId] = _allNfts.length;
    _allNfts.push(tokenId);
}

function _addTokenToOwnerEnumeration(address to, uint tokenId) private {
    uint lenght = ERC721.balanceOf(to);
    _ownedTokens[to][lenght] = tokenId;
    _idToOwnedIndex[tokenId] = lenght;
}

function _removeTokenFromOwnerEnumeration(address from, uint tokenId)
private {
    uint lastTokenIndex = ERC721.balanceOf(from) - 1;
    uint tokenIndex = _idToOwnedIndex[tokenId];

    if (tokenIndex != lastTokenIndex) {
        uint lastTokenId = _ownedTokens[from][lastTokenIndex];
        _ownedTokens[from][tokenId] = lastTokenId;
        _idToOwnedIndex[lastTokenId] = tokenIndex;
    }
}

```

```

delete _idToOwnedIndex[tokenId];
delete _ownedTokens[from][lastTokenIndex];
}

```

```

function _removeTokenFromAllTokensEnumeration(uint tokenId) private {
uint lastTokenIndex = _allNfts.length - 1;
uint tokenIndex = _idToNftIndex[tokenId];
uint lastTokenId = _allNfts[lastTokenIndex];

```

```

_allNfts[tokenIndex] = lastTokenId;
_idToNftIndex[lastTokenId] = tokenIndex;

```

```

delete _idToNftIndex[tokenId];
_allNfts.pop();
}
}

```

components/Providers/account/index.tsx:

```

import { useAccount } from "@hooks";
import { useRouter } from "next/router";
import { createContext } from "react";
import { toast } from "react-toastify";

```

```

interface AccountLockerProps {
children: React.ReactNode
}

```

```

const Web3Context = createContext({});

```

```

export default function AccountLocker({ children }: AccountLockerProps) {
  const { account } = useAccount();
  const router = useRouter();
  const path = router.asPath;
  const root = [
    '/',
    '/reports',
    '/strategy',
    '/privacy-policy'
  ];

  const allowed = [
    '/about-us'
  ]

  if (((!account.data    &&    !account.isLoading)    ||    account.error)
    && !root.includes(path) && !allowed.find((item) => path.includes(item))) {
    router.push('/login');
  }

  return children
}

```

components/Providers/web3/index.tsx:

```

import { createContext, useContext, useEffect, useState } from "react";
import { Web3State, createDefaultState, createWeb3State, loadContract } from
"./utils";

import { providers } from 'ethers';
import { setupHook } from "@hooks";
import { MetaMaskInpageProvider } from "@metamask/providers";

```

```

import { NftMarketContract } from "@types/nftMarketContract";

function pageReload() {
  window.location.reload()
}

const handleAccount = (ethereum: MetaMaskInpageProvider) => async () => {
  const isLocked = !(await ethereum._metamask.isUnlocked());
  if (isLocked) {
    pageReload();
  }
}

const setGlobalListeners = (ethereum: MetaMaskInpageProvider) => {
  ethereum.on("chainChanged", pageReload);
  ethereum.on("accountsChanged", handleAccount(ethereum));
}

const removeGlobalListeners = (ethereum: MetaMaskInpageProvider) => {
  ethereum?.removeListener("chainChanged", pageReload);
  ethereum?.removeListener("accountsChanged", handleAccount)
}

const Web3Context = createContext<Web3State>(createDefaultState());

interface Web3ProviderProps {
  children: React.ReactNode
}

export default function Web3Provider({ children }: Web3ProviderProps) {

```

```

const [web3Api, setWeb3Api] = useState<Web3State>(createDefaultState());

useEffect(() => {
  async function initWeb3() {
    try {
      const provider = new providers.Web3Provider(window.ethereum as any);
      const contract = await loadContract("NftMarket", provider);
      console.log(contract)

      const signer = provider.getSigner();
      const signedContract = contract.connect(signer);

      setGlobalListeners(window.ethereum);
      setWeb3Api(createWeb3State({
        ethereum: window.ethereum,
        provider,
        contract: signedContract as unknown as NftMarketContract,
        isLoading: false
      })))
    } catch (e: any) {
      console.log(e)
      setWeb3Api((api) => createWeb3State({
        ...api as any,
        isLoading: false
      })))
    }
  };
  initWeb3();
  return () => removeGlobalListeners(window.ethereum)
}, [])

```

```

return (
  <Web3Context.Provider value={web3Api}>
    {children}
  </Web3Context.Provider>
)
}

```

```

export function useHooks() {
  const { hooks } = useWeb3();
  return hooks
}

```

```

export function useWeb3() {
  return useContext(Web3Context)
}

```

components/Providers/web3/utils.ts:

```

import { Web3Hook, setupHook } from
'@/components/Hooks/Web3/setupHooks';
import { Web3Deps } from '@_types/hook';
import { MetaMaskInpageProvider } from '@metamask/providers';
import { Contract, ethers, providers } from 'ethers';

declare global {
  interface Window {
    ethereum: MetaMaskInpageProvider;
  }
}

```

```

export type Web3Params = {
  ethereum: MetaMaskInpageProvider | null;
  provider: ethers.providers.Web3Provider | null;
  contract: Contract | null;
}

type Nullable<T> = {
  [P in keyof T]: T[P] | null
}

```

```

export type Web3State = {
  isLoading: boolean;
  hooks: Web3Hook;
} & Nullable<Web3Deps>;

```

```

export const createDefaultState = () => {
  return {
    ethereum: null,
    provider: null,
    contract: null,
    isLoading: true,
    hooks: (setupHook({isLoading: true} as any))
  }
}

```

```

export const createWeb3State = ({
  ethereum, provider, contract, isLoading
}: Web3Deps) => {
  return {
    ethereum,
    provider,

```

```

contract,
isLoading: false,
hooks: setupHook({ethereum, provider, contract, isLoading})
}
}

const NETWORK_ID = process.env.NEXT_PUBLIC_NETWORK_ID;

export const loadContract = async (name: string, provider:
providers.Web3Provider): Promise<Contract> => {
  if (!NETWORK_ID) {
    return Promise.reject('Network ID is not defined!');
  }

  const res = await fetch (`/contracts/${name}.json`);
  const Artifact = await res.json();
  console.log(Artifact)

  if (Artifact.networks[NETWORK_ID].address) {
    const contract = new ethers.Contract(
      Artifact.networks[NETWORK_ID].address,
      Artifact.abi,
      provider
    )
    return contract
  } else {
    return Promise.reject(`Contract: [${name}] cannot be loaded!`);
  }
}

```

components/Hooks/Web3/index.tsx:

```
import { useHooks } from "@providers/web3"
```

```
export const useAccount = () => {  
  const hooks = useHooks();  
  const swrRes = hooks.useAccount();  
  return {  
    account: swrRes  
  }  
}
```

```
export const useNetwork = () => {  
  const hooks = useHooks();  
  const swrRes = hooks.useNetwork();  
  return {  
    network: swrRes  
  }  
}
```

```
export const useListedNfts = () => {  
  const hooks = useHooks();  
  const swrRes = hooks.useListedNfts();  
  return {  
    nfts: swrRes  
  }  
}
```

```
export const useOwnedNfts = () => {  
  const hooks = useHooks();  
  const swrRes = hooks.useOwnedNfts();
```

```
return {  
  nfts: swrRes  
}  
}
```

components/Hooks/Web3/setupHooks.ts:

```
import { Web3Deps } from "@_types/hook";  
import { hookFactory as createAccountHook, UseAccountHook } from  
"./useAccount";  
import { hookFactory as createNetworkHook, UseNetworkHook } from  
"./useNetwork";  
import { hookFactory as createListedNftsHook, UseListedNftsHook } from  
"./useListedNfts";  
import { hookFactory as createOwnedNftsHook, UseOwnedNftsHook } from  
"./useOwnedNfts";
```

```
export type Web3Hook = {  
  useAccount: UseAccountHook;  
  useNetwork: UseNetworkHook;  
  useListedNfts: UseListedNftsHook;  
  useOwnedNfts: UseOwnedNftsHook;  
}
```

```
export type SetupHook = {  
  (d: Web3Deps): Web3Hook  
}
```

```
export const setupHook: SetupHook = (deps) => {  
  return {  
    useAccount: createAccountHook(deps),  
    useListedNfts: createListedNftsHook(deps),
```

```

useNetwork: createNetworkHook(deps),
useOwnedNfts: createOwnedNftsHook(deps)
}
}

```

components/Hooks/Web3/useAccount.ts:

```

    'use client'

import { CryptoHookFactory } from "@_types/hook";
import { useEffect, useState } from "react";
import { toast } from "react-toastify";
import useSWR from "swr";

//
type UseAccountResponse = {
  connect: () => void;
  isLoading: boolean;
  isInstalled: boolean;
}

type AccountHookFactory = CryptoHookFactory<string, UseAccountResponse>

export type UseAccountHook = ReturnType<AccountHookFactory>

export const hookFactory: AccountHookFactory = ({ provider, ethereum,
isLoading }) => () => {
  const { data, mutate, isValidating, ...swr } = useSWR(
    provider ? "web3/useAccount" : null,
    async () => {
      const accounts = await provider!.listAccounts();
      const account = accounts[0];
    }
  );

```

```

if (!account) {
  throw "Cannot retrieve account! Please, connect to web3 wallet."
}
return account
},
{
  revalidateOnFocus: false,
  shouldRetryOnError: false
}
)

useEffect(() => {
  ethereum?.on("accountsChanged", handleAccountsChanged)
  return () => {
    ethereum?.removeListener("accountsChanged", handleAccountsChanged)
  }
})

const handleAccountsChanged = (...args: unknown[]) => {
  const accounts = args[0] as string[];
  if (!accounts.length) {
    throw "Cannot retrieve account! Please, connect to web3 wallet."
  } else if (accounts[0] !== data) {
    mutate(accounts[0]);
  }
}

const connect = async () => {
  try {
    ethereum?.request({ method: "eth_requestAccounts" })
  }
}

```

```

    } catch (e) {
      console.log(e)
    }
  }

  return {
    ...swr,
    data,
    isValidating,
    isLoading: isLoading as boolean,
    isInstalled: ethereum?.isMetaMask || false,
    mutate,
    connect
  };
}

```

```

export const useAccount = hookFactory({ ethereum: undefined, provider:
undefined })

```

components/Hooks/Web3/useListedNfts.ts:

```

import { CryptoHookFactory } from "@_types/hook";
import { Nft } from "@_types/nft";
import { ethers } from "ethers";
import useSWR from "swr";

type UseListedNftsResponse = {
  buyNft: (tokenId: number, value: number) => Promise<void>
}

type ListedNftsHookFactory = CryptoHookFactory<Nft[],
UseListedNftsResponse>

```

```

export type UseListedNftsHook = ReturnType<ListedNftsHookFactory>

export const hookFactory: ListedNftsHookFactory = ({contract}) => () => {
  const {data, ...swr} = useSWR(
    contract ? "web3/useListedNfts" : null,
    async () => {
      const nfts = [] as Nft[];
      const coreNfts = await contract!.getAllNftsOnSale();

      for (let i = 0; i < coreNfts.length; i++) {
        const item = coreNfts[i];
        const tokenURI = await contract!.tokenURI(item.tokenId);
        const metaRes = await fetch(tokenURI);
        const meta = await metaRes.json();

        nfts.push({
          price: parseFloat(ethers.utils.formatEther(item.price)),
          tokenId: item.tokenId.toNumber(),
          creator: item.creator,
          isListed: item.isListed,
          meta: meta
        })
      }

      return nfts;
    }
  )

  const buyNft = async (tokenId: number, value: number) => {

```

```

    try {
      await contract?.buyNft(tokenId, {value:
ethers.utils.parseEther(value.toString())});
    } catch (e: any) {
      console.log(e.message)
    }
  }
  return {
    ...swr,
    buyNft,
    data: data || [],
  };
}

```

/components/Hooks/Web3/useNetwork.ts;

```

import { CryptoHookFactory } from "@_types/hook";
import { useEffect } from "react";
import useSWR from "swr";

```

```

const NETWORKS: {[k:string]: string} = {
  1: "Ethereum Main Network",
  3: "Ropsten Test Network",
  4: "Rinkeby Test Network",
  5: "Goerli Test Network",
  42: "Kovan Test Network",
  56: "Binance Smart Chain",
  1337: "Ganache",
}

```

```

const targetId = process.env.NEXT_PUBLIC_TARGET_CHAIN_ID as string;

```

```

const targetNetwork = NETWORKS[targetId];

type UseNetworkResponse = {
  isLoading: boolean;
  isSupported: boolean;
  targetNetwork: string;
}

type NetworkHookFactory = CryptoHookFactory<string, UseNetworkResponse>

export type UseNetworkHook = ReturnType<NetworkHookFactory>

export const hookFactory: NetworkHookFactory= ({provider, isLoading}) => ()
=> {
  const {data, isValidating, ...swr} = useSWR(
    provider ? "web3/useNetwork" : null,
    async () => {
      const chainId = (await provider!.getNetwork()).chainId;
      if(!chainId) {
        throw "Cannot retrieve network. Please, refresh browser or connect to other one."
      }

      return NETWORKS[chainId]
    },
    {
      revalidateOnFocus: false
    }
  )

  return {

```

```

...SWR,
data,
isValidating,
isLoading: isLoading || isValidating,
targetNetwork,
isSupported: data === targetNetwork
};
}

```

components/Hooks/Web3/useOwnedNfts.ts:

```

import { CryptoHookFactory } from "@_types/hook";
import { Nft } from "@_types/nft";
import { ethers } from "ethers";
import useSWR from "swr";

type UseOwnedNftsResponse = {}

type OwnedNftsHookFactory = CryptoHookFactory<Nft[],
UseOwnedNftsResponse>

export type UseOwnedNftsHook = ReturnType<OwnedNftsHookFactory>

export const hookFactory: OwnedNftsHookFactory = ({contract}) => () => {
  const {data, ...swr} = useSWR(
    contract ? "web3/useOwnedNfts" : null,
    async () => {
      const nfts = [] as Nft[];
      const coreNfts = await contract!.getOwnedNfts();

      for (let i = 0; i < coreNfts.length; i++) {
        const item = coreNfts[i];
        const tokenURI = await contract!.tokenURI(item.tokenId);

```

```

const metaRes = await fetch(tokenURI);
const meta = await metaRes.json();

nfts.push({
  price: parseFloat(ethers.utils.formatEther(item.price)),
  tokenId: item.tokenId.toNumber(),
  creator: item.creator,
  isListed: item.isListed,
  meta: meta
})
}

return nfts;
}
)
return {
  ...swr,
  data: data || [],
};
}

```