Blockchain Application Development CSE 526
Project - Phase III

Title

BookChain - NFT for Books

Team Members

First Name	Ashwin Kumar	Mohit Sai Aravind
Last Name	Jayakumar	Nunna
Email	jayakuma@buffalo.edu	mnunna@buffalo.edu
Person Number	50469087	50468322

Issues Addressed

Books are an integral part of our lives. Reading was a common hobby among the past generation. People used to visit bookstores and libraries to buy books back in those days. With the fast moving world, people are no longer interested in visiting bookstores for books that are readily available in digital world. But there are reading enthusiasts who might be interested in books of the past generation which may not be available in market today. There might be only a few of these available around the world in random places. These can be of use to many of the reading enthusiasts around the world. This can be solved with a decentralized blockchain technology.

Abstract

The present generation are not only interested in the fictions and fantasies of recent times but also the novels and dramas of the past. Marketplace for book NFTs of the past as well as the current generation will bridge the generational gap and cater to the needs of all generations. With the help of NFTs, we will be able to discover more rare books of this kind.

Our application creates unique NFTs for every book. Each NFT is uniquely identified by a UID : Unique Identification Number. NFTs in the book marketplace could provide a means for tracing the ownership of the NFT. This enables us to trace back the various owners and find out the history of the book from the immutable blockchain ledger. This also ensures transparent transfer of book NFTs among users.

The transfer of NFTs also involves cryptocurrency, which act as digital tokens for NFT purchase. This could appeal to the younger generation, who are more into cryptocurrency and would be fascinated by the NFT collectibles in this digital world.

The price of NFTs are also determined by various factors such as edition, release year, genre, book format and limited edition copies etc. This way, we can dynamically define the price of a book to be fair to both the owner as well as the buyer of the Book NFT. These NFTs can be bought with Bookies, the ERC20 tokens used during NFT transactions.

Digital Asset and Token

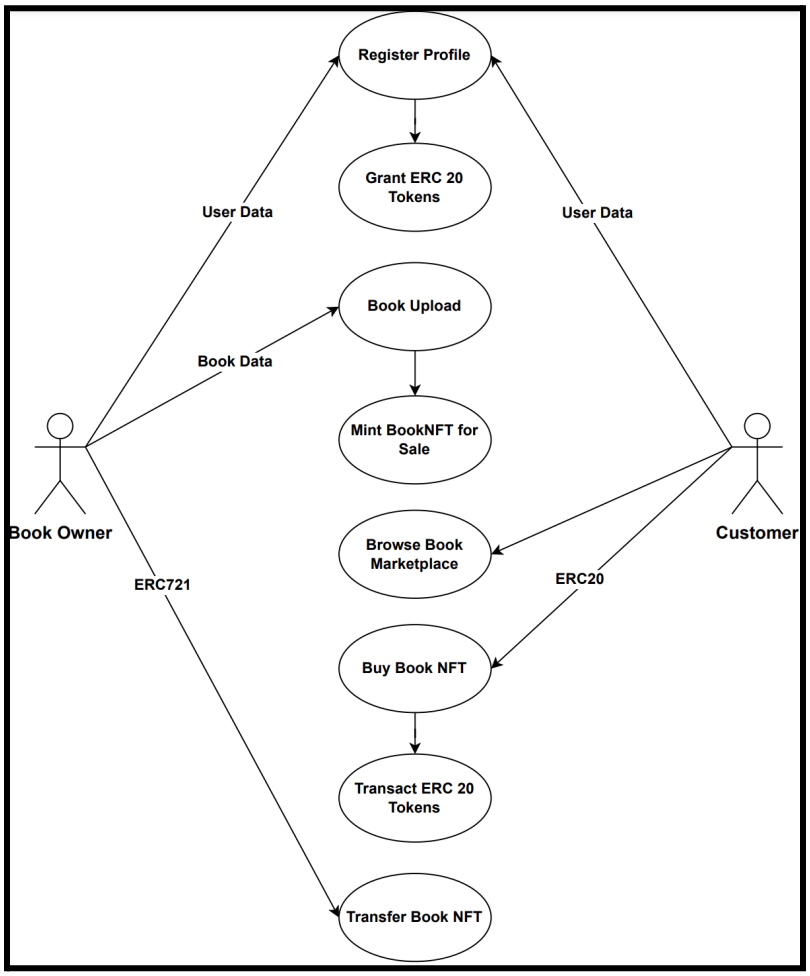
Token : Bookies
Digital asset : BookNFT

The token is named **Bookies** representing the **ERC20** Book Tokens used to make transaction during purchase of NFT. The digital asset is named **BookNFT** representing the **ERC721** tokens which is unique for every **book** on marketplace.

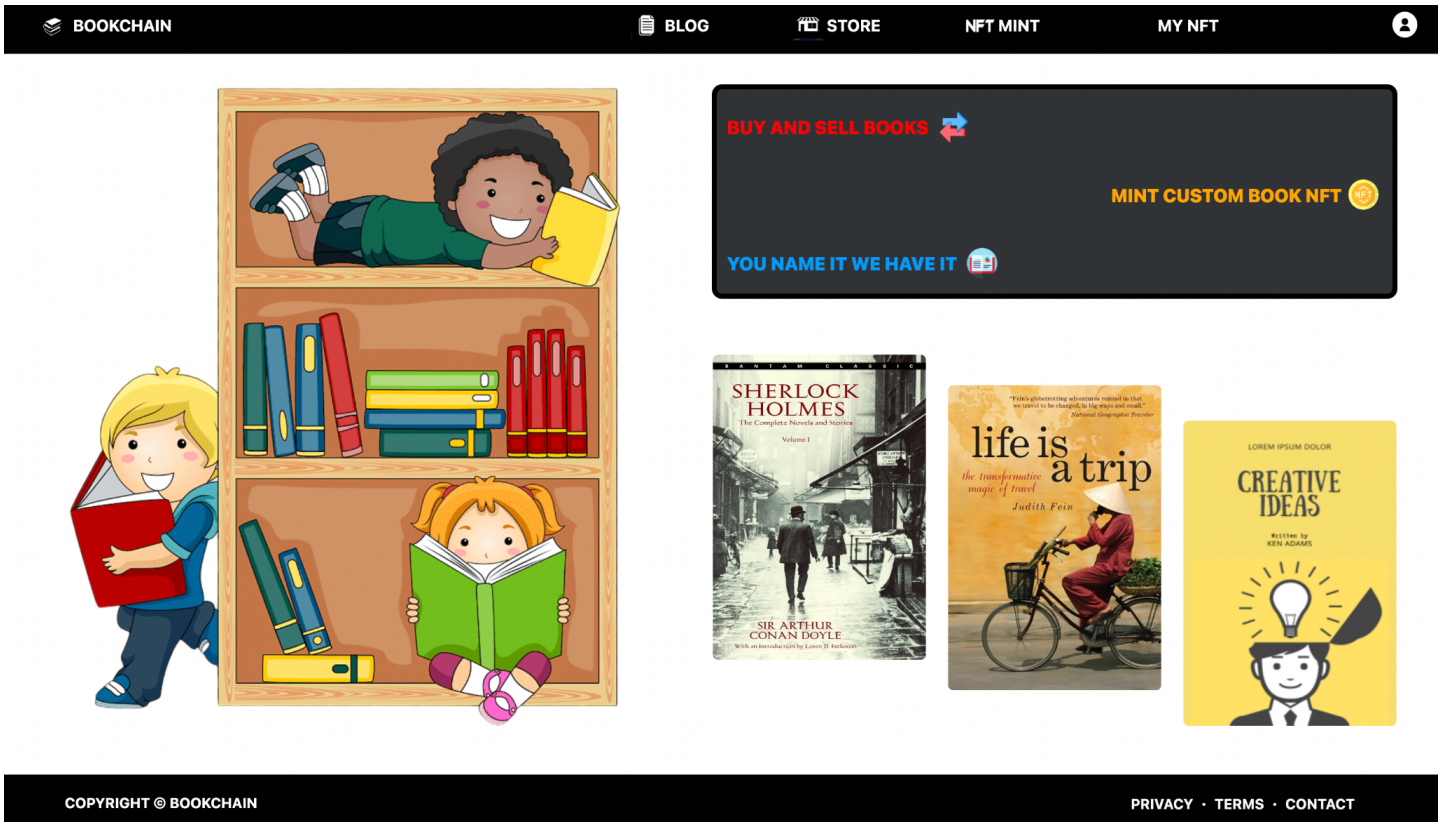
Reasoning

Book is the asset around which the application functions. Every book has a unique identification number **UID**. Hence this is used as the unique identifier to create NFTs for each book. **ERC721** is the commonly standard used to mint unique NFT for each unique asset and avoid duplication. **ERC20** is the commonly used to standard to generate tokens for transaction.

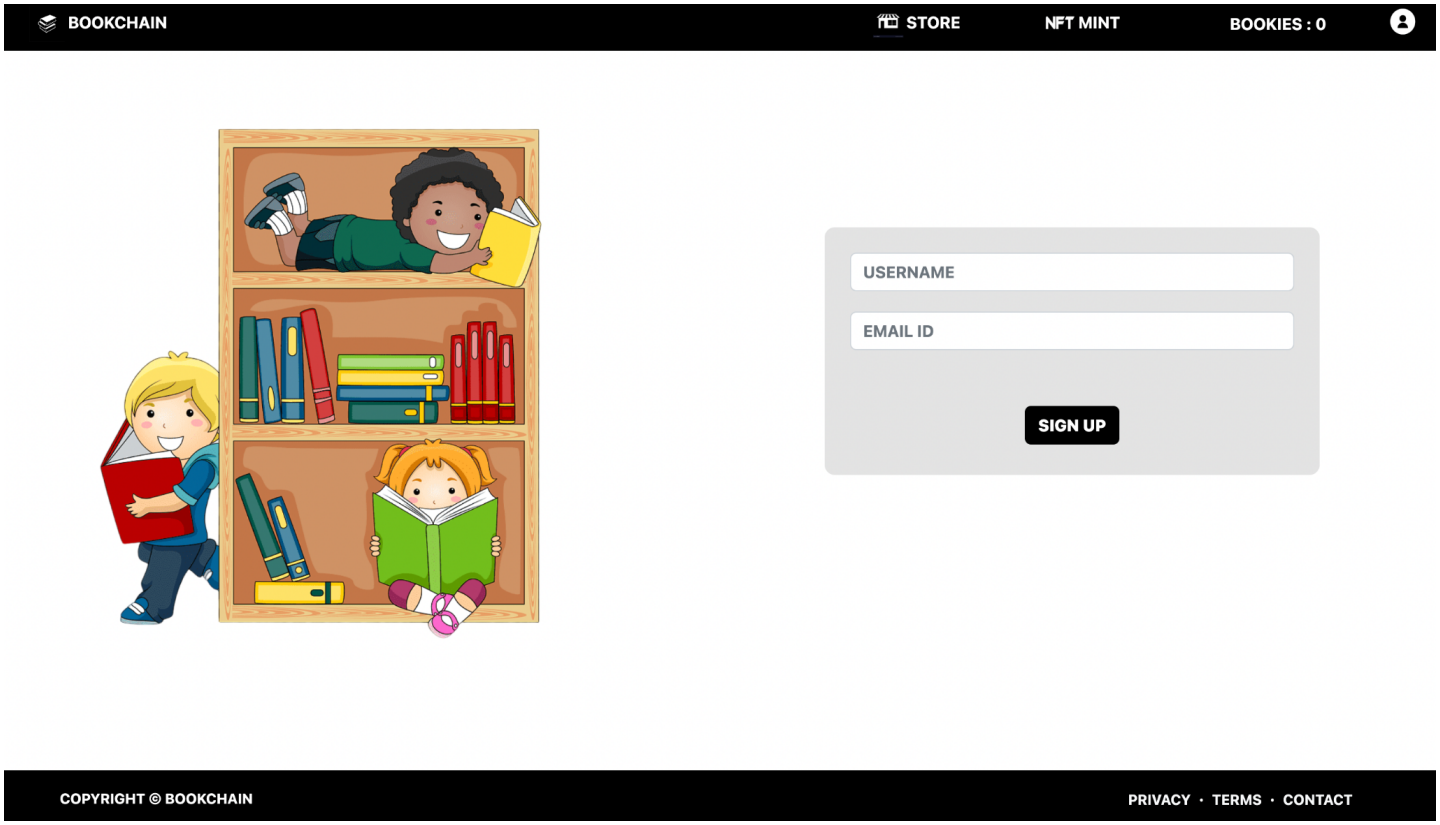
Use Case Diagram



Home Page



User Signup



NFT Mint

BOOKCHAIN | STORE | NFT MINT | MY NFT | BOOKIES : 1000

CREATE BOOK NFT !!!

BOOK TITLE | AUTHOR

BOOK ID | PAGES | EDITION | RELEASE YEAR

GENRE | FORMAT | SPECIAL EDITION

DESCRIPTION

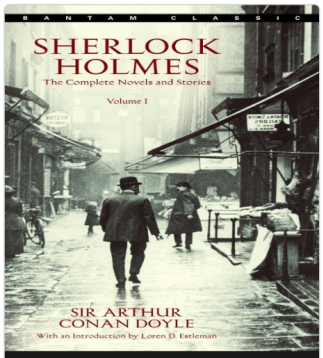
MINT NFT

COPYRIGHT © BOOKCHAIN | PRIVACY · TERMS · CONTACT

Book NFT

BOOKCHAIN | STORE | NFT MINT | MY NFT | BOOKIES : 1000

EXPLORE YOUR BOOK COLLECTION



BOOK : SHERLOCK HOLMES : COMPLETE SERIES

AUTHOR : ARTHUR CONAN DOYLE

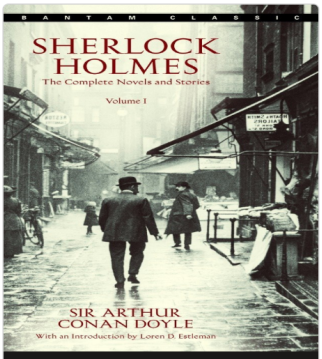
INFO

COPYRIGHT © BOOKCHAIN | PRIVACY · TERMS · CONTACT

Book Marketplace

BOOKCHAIN | STORE | NFT MINT | MY NFT | BOOKIES : 1000

EXPLORE YOUR BOOK COLLECTION



BOOK : SHERLOCK HOLMES : COMPLETE SERIES

AUTHOR : ARTHUR CONAN DOYLE

INFO

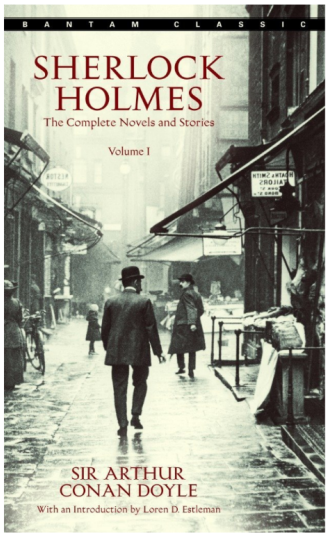
COPYRIGHT © BOOKCHAIN

PRIVACY · TERMS · CONTACT

Book Info

BOOKCHAIN | STORE | NFT MINT | MY NFT | BOOKIES : 1000

BOOK INFO



SHERLOCK HOLMES : COMPLETE SERIES

ADVENTURES OF SHERLOCK HOLMES AND HIS FRIEND DR. WATSON.

UID : 1234

AUTHOR : ARTHUR CONAN DOYLE

EDITION : 1

PAGES : 500

RELEASE YEAR : 2015

GENRE : Adventure

FORMAT : Hardcover

SPECIAL EDITION : Yes

MARKET PRICE : 180

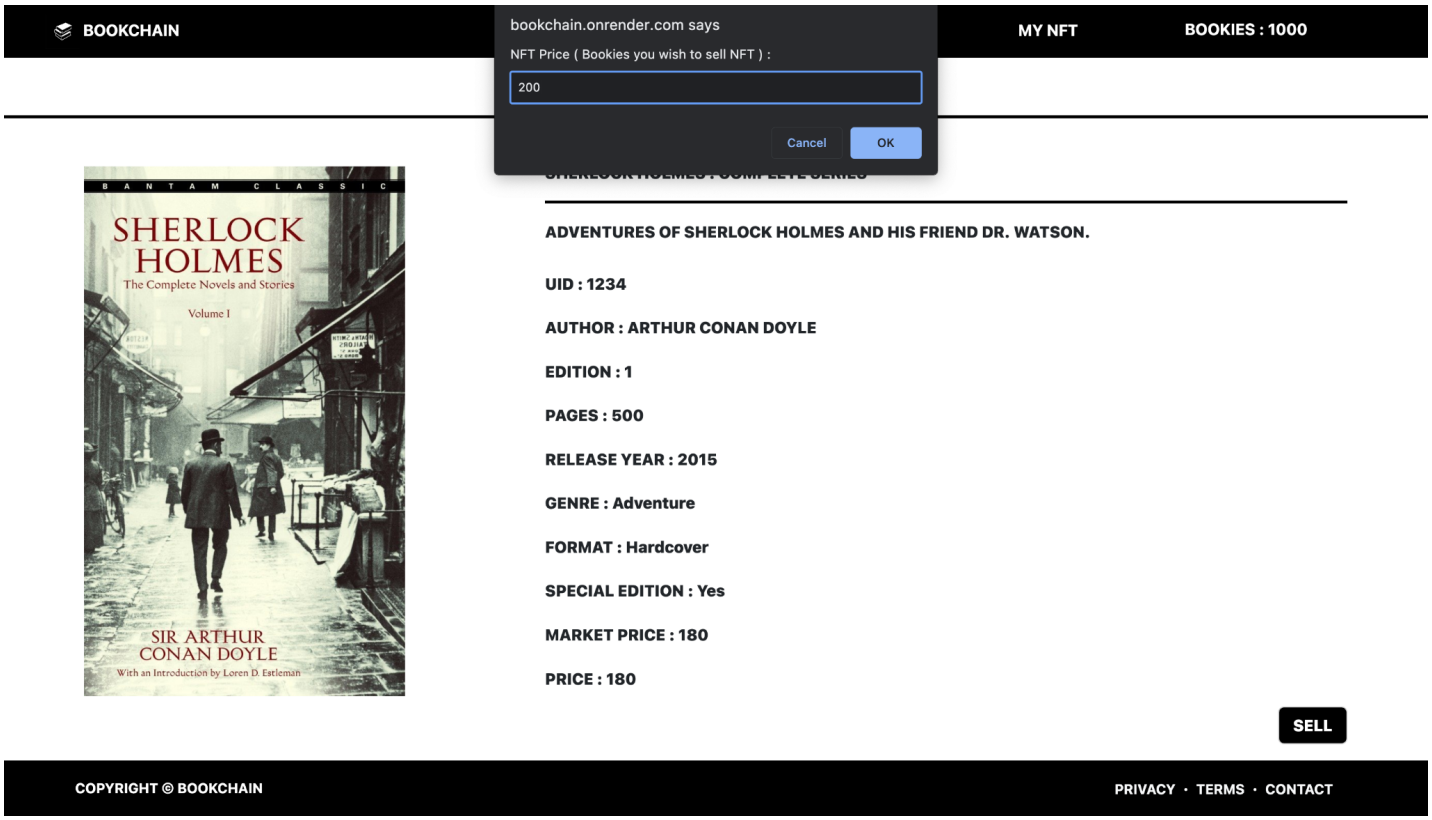
PRICE : 180

SELL

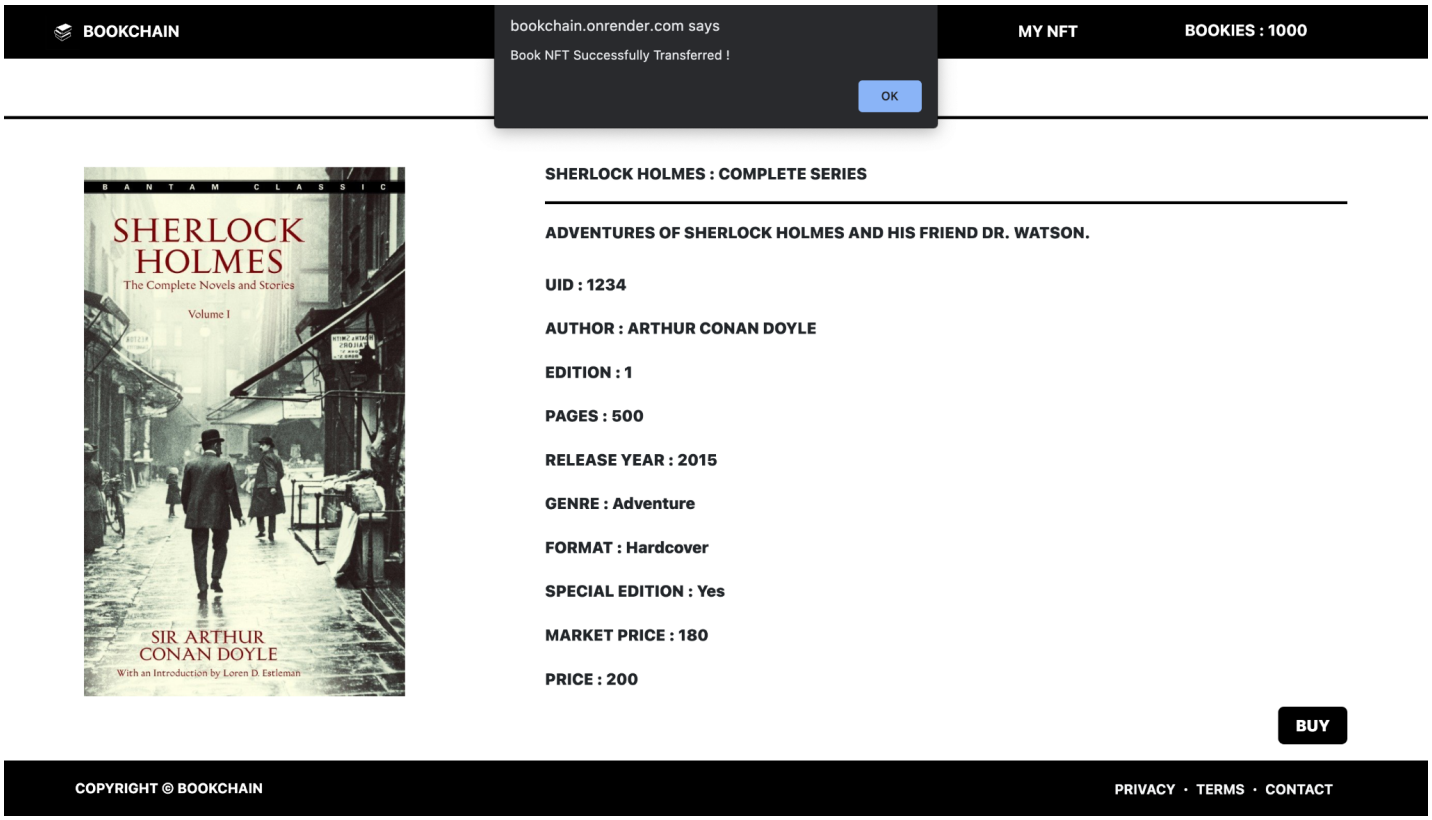
COPYRIGHT © BOOKCHAIN

PRIVACY · TERMS · CONTACT

Sell Book



Buy Book



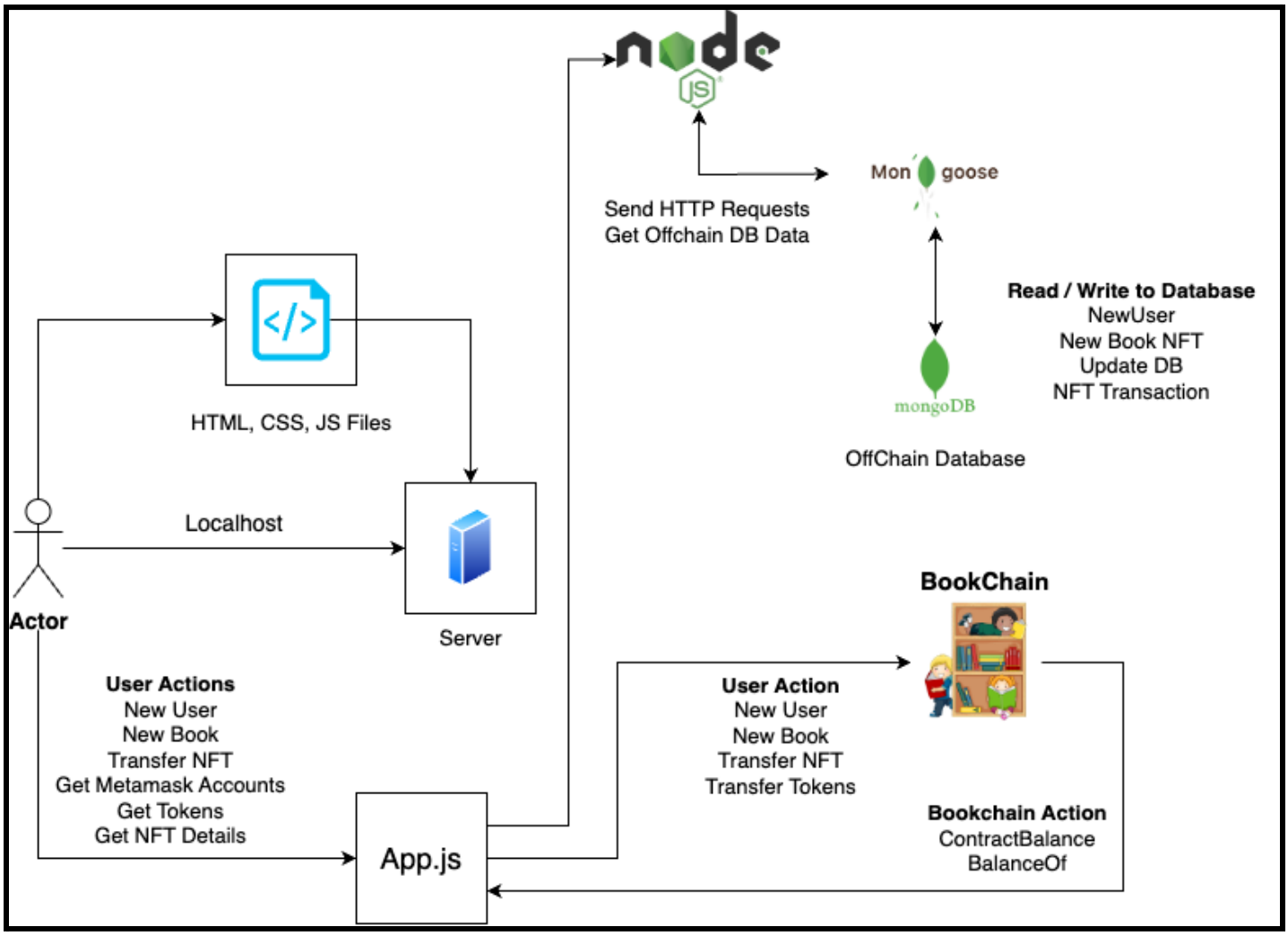
Contract Diagram

SMART CONTRACT BOOKCHAIN
+ Bookies token + BookNFT nft + struct Book + struct User + mapping (address => User) UserTable + mapping (address => uint256 [ ]) UserBookTable + mapping (uint256 => Book) BookTable
+ event UserSignup(address UserAddress, string username, string message) + event BookMint(address UserAddress, uint256 bookid, string message) + event NftTransfer(address buyer, address owner, uint256 bookid, string message)
+ modifier NotOwner(address buyer, address owner) + modifier ValidUser(address UserAddress) + modifier ExistingUser(address UserAddress) + modifier SufficientFunds(address UserAddress, uint256 PriceNFT)
+ Constructor(TokenAddress, NFTAddress) + Function NewUser(address UserAddress, string memory username, string memory email) public ExistingUser(UserAddress) { } + Function NewBook(address UserAddress ,string memory title, string memory author, string memory description, uint256 edition, uint256 pages, uint256 uid) public ValidUser (UserAddress) { } + Function BuyNFT(address payable owner, address payable buyer, uint256 TokenId, uint256 PriceNFT) public SufficientFunds(buyer, PriceNFT) { } + Function UserBalance() view public returns(uint) { } + Function ContractBalance() view public returns(uint) { }

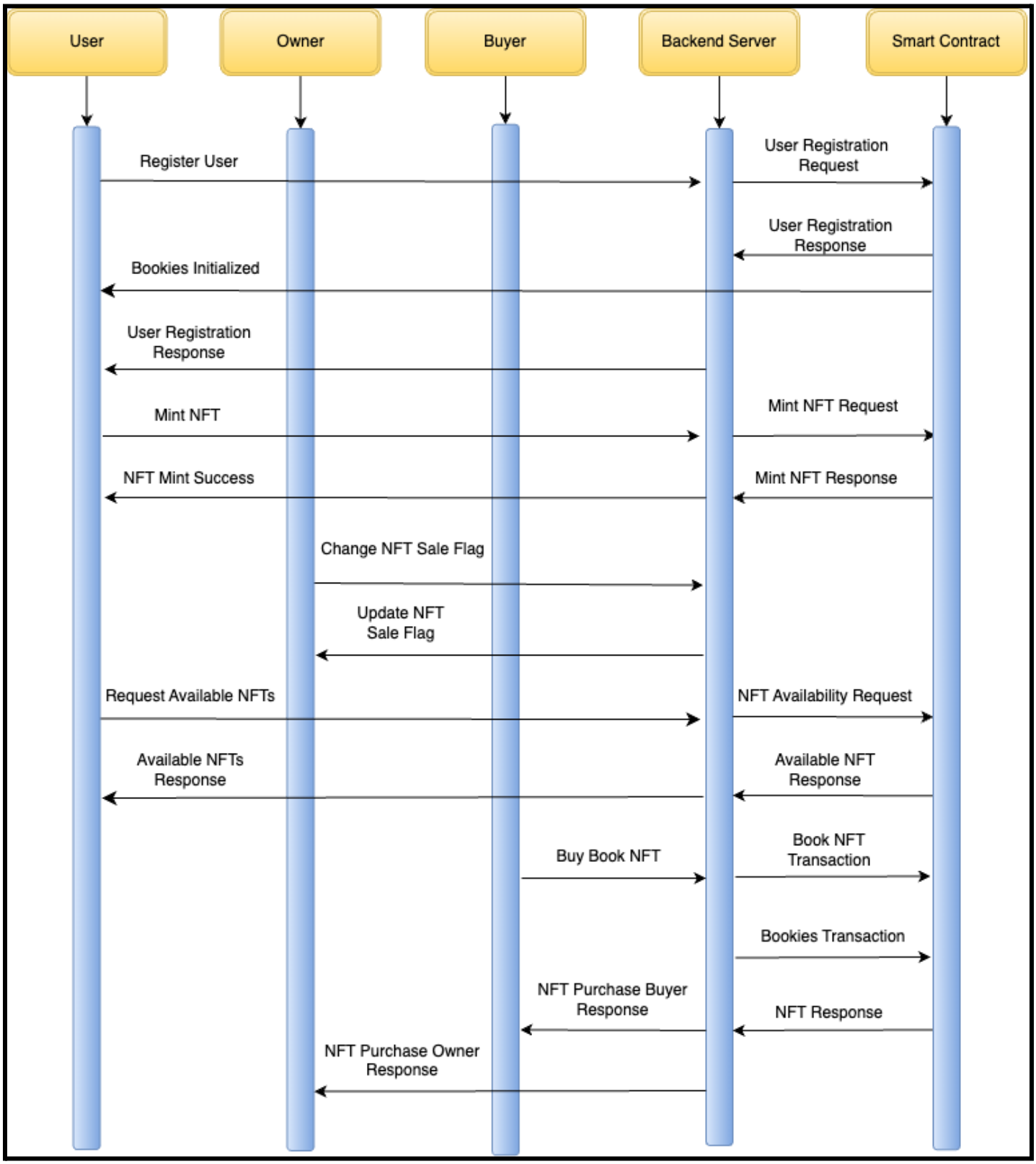
SMART CONTRACT BOOKIES
< No Data Definition for this contract >
< No Events for this contract >
< No Modifier for this contract >
+ Constructor( ) ERC20("Bookies", "BK"){ } + GenerateToken(address SmartContract) + TransferTokens(address from, address to, uint256 Token) public { }

SMART CONTRACT BOOKNFT
< No Data Definition for this contract >
< No Events for this contract >
< No Modifier for this contract >
+ Constructor ( ) ERC721("BookNFT", "BFT") { } + MintNFT(address user, address SmartContract, uint256 uid) public { } + TransferNFT(address from, address to, uint256 TokenId, address SmartContract) public { }

Architectural Diagram



Sequence Diagram



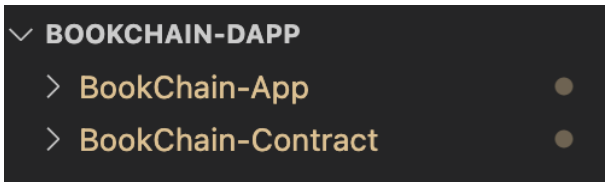
Deployment Steps (Ganache Network)

We develop 3 smart contracts for our application

1. BookChain : **Application** Smart Contract
2. BookToken : Smart Contract implementing **ERC20** tokens for NFT Transactions
3. BookNFT : Smart Contract implementing **ERC721** tokens to mint Book NFT

We have a **Bookchain-Dapp**

1. **BookChain-App** : Application code implementing the **Web3 Application**
2. **BookChain-Contract**: Smart Contract Solidity code implementing the **Non Fungible Tokens**



Run **Ganache Ethereum** server on local system to run and test our BookChain application. This setup is used to import accounts for application testing purposes.

Move into the **BookChain-Contract** and run **truffle compile** to compile the contracts and make them ready for deployment.

```
(base) ashwin@Ashwins-MBP BookChain-Contract % truffle compile

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.
(base) ashwin@Ashwins-MBP BookChain-Contract %
```

Once compilation is done, perform **truffle migrate --reset** to deploy the contracts. On deployment, each contract has a unique contract address.

```
Replacing 'Bookchain'
---------------------
> transaction hash:    0x7ee111afdfcd8899d21f9a5e64ed1ab9a1092d49a5322c8ea5a2ffcaf54b70f3
> Blocks: 0            Seconds: 0
> contract address:    0xD9d8334c5B00716Bb814eFC906c7e5c6ABD33Be2
> block number:        260
> block timestamp:     1682901539
> account:             0x80C61249Ca76CA6eCA8435EA4F66d1636B598c31
> balance:             99.311729482015195662
> gas used:            1946206 (0x1db25e)
> gas price:           2.500000008 gwei
> value sent:          0 ETH
> total cost:          0.004865515015569648 ETH

> Saving artifacts
-------------------------------------
> Total cost:     0.013405065042896208 ETH
```

```
Summary
=======
> Total deployments:   3
> Final cost:          0.013405065042896208 ETH


(base) ashwin@Ashwins-MBP BookChain-Contract % truffle migrate --reset
```

Move into **BookChain-App** and run **npm install** to install the required packages and then run **npm start** to listen on the port 3000 specified in index.js.

```
(base) ashwin@Ashwins-MBP BookChain-App % npm start

> bookchain-app@1.0.0 start
> node index.js

App connected on port 3000!
Connected to MongoDB!
```

Hit the url http://localhost:3000/ in browser and you will be greeted with the landing page of our Bookchain Web Application.

We have a **Bookchain-Dapp**

1. **BookChain-App** : Application code implementing the **Web3 Application**
2. **BookChain-Contract**: Smart Contract Solidity code implementing the **Non Fungible Tokens**

On Chain Data

We have 3 smart contracts for our application

1. BookChain : **Application** Smart Contract
2. Bookies : Smart Contract implementing **ERC20** tokens for NFT Transactions
3. BookNFT : Smart Contract implementing **ERC721** tokens to mint Book NFT

Smart Contract Functions

constructor(address TokenAddress, address NFTAddress)

This is used to initialize the Bookies and BookNFT contract variables to be used in the Bookchain contract and takes the contract address of Bookies and BookNFT as input.

function NewUser(address UserAddress, string memory username, string memory userId)

This function is used to create a profile for every user on our application. The function takes as input the user metamask wallet address, username and userId, object id of the user data object stored in Mongo DB.

function NewBook(address UserAddress, uint256 bookid, string memory bookname, string memory bookId)

This function is used to create a Book NFT for a new book created by the user. This function takes as input the metamask wallet address of the user creating the Book NFT, bookid unique for every book used to mint the Book NFT, name of the book and bookId, object id of the book data object stored in Mongo DB.

function BuyNFT(address payable owner, address payable buyer, uint256 bookid, uint256 PriceNFT)

This function is used to transfer ownership of NFT from owner to buyer of Book NFT. This function takes as input metamask wallet address of Book NFT owner, metamask wallet address of Book NFT buyer, bookid unique for every Book NFT and the value of the Book NFT, which is transferred from buyer to owner account in Bookies (ERC 20 Tokens).

function UserBalance(address UserAddress)

This function returns the number of Bookies (ERC 20 Tokens) available in a user account and takes user metamask wallet address as input.

function ContractBalance()

This function returns the number of Bookies (ERC 20 Tokens) available in the BookChain smart contract.

Modifiers

modifier NotOwner(address buyer, address owner)

This modifier is used to restrict the owner of the Book NFT from buying his own NFT.

modifier ValidUser(address UserAddress)

This modifier is used to verify whether a registered user is creating a Book NFT.

modifier ExistingUser(address UserAddress)

This modifier is used to restricting an existing user from creating a new profile with same wallet address.

modifier SufficientFunds(address UserAddress, uint256 PriceNFT)

This modifier is used to verify whether the user buying the Book NFT has sufficient funds to buy the Book NFT.

Data Structures

```
struct Book {
    address owner;
    string bookname;
    uint256 bookid;
    string bookId;
  }
```

This structure stores data like owner of Book NFT, name of book, unique bookid of each Book NFT.

```
struct User {
    string username;
    string userId;
    uint256 flag;
  }
```

This structure stores data like username, userId and flag set to 1 of each user profile in the application.

Mapping Structures

mapping (address => User) public UserTable

This stores a mapping of each user address to a User structure storing user profile details.

mapping (address => uint256[]) public UserBookTable

This stores a mapping of user address to a list of bookids denoting the id of the Book NFT.

mapping (uint256 => Book) public BookTable

This stores a mapping of unique bookid to a Book structure storing the details of Book NFT minted.

Events

event UserSignup(address UserAddress, string username, string message)

This event will be triggered when a user is successfully signed up in the application and a user profile is created.

event BookMint(address UserAddress, uint256 bookid, string message)

This event will be triggered when a new book is created in the application and a Book NFT is minted for the book.

event NftTransfer(address buyer, address owner, uint256 bookid, string message)

This event will be triggered when a Book NFT is successfully transferred from owner to buyer and Bookies (ERC 20 Tokens) are successfully transferred from buyer to owner.

Off Chain Data

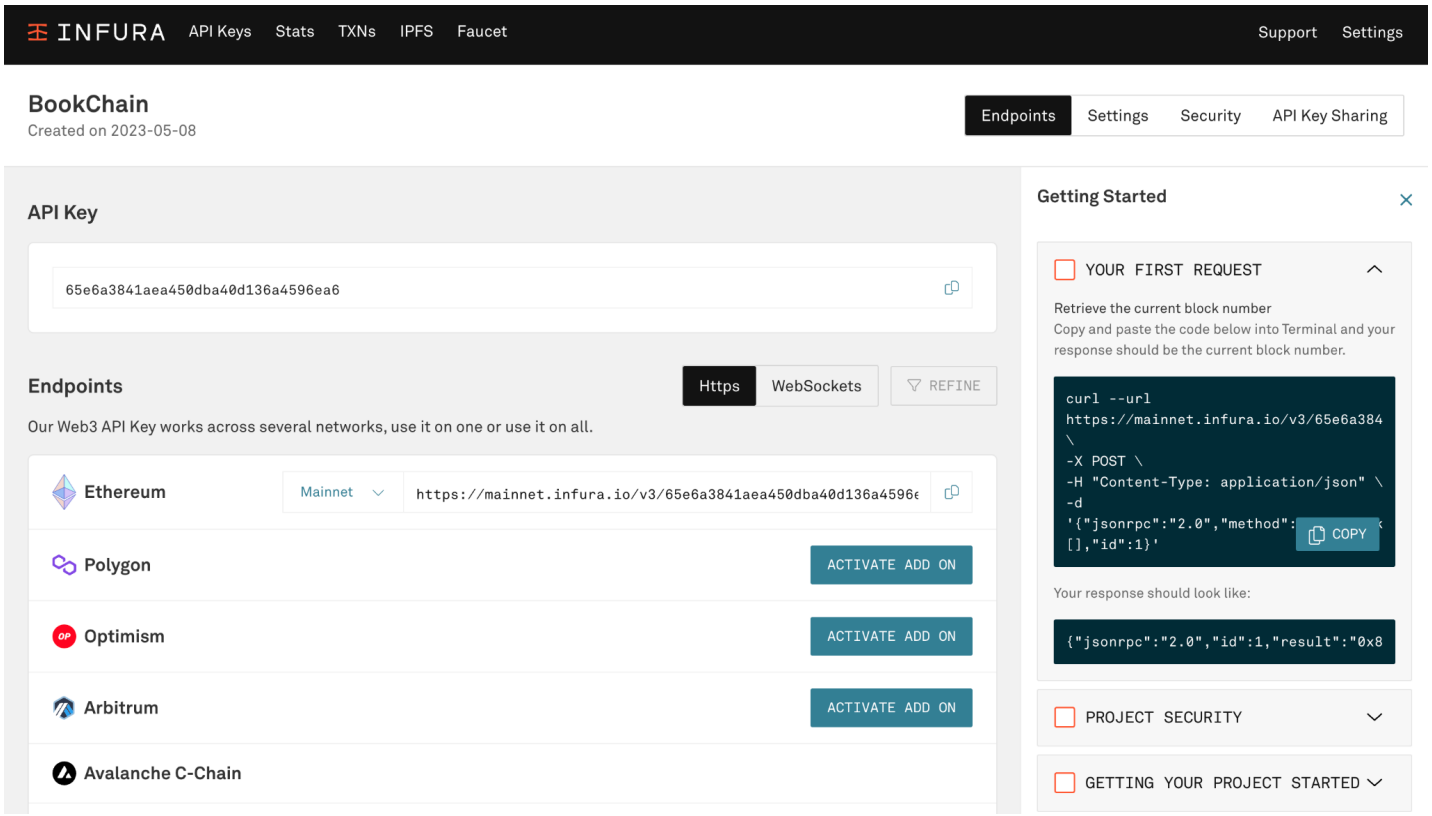
The off chain data is stored in Mongo DB.

UserDetails	
metamask	string
username	string
email	string
nft	array [string]

BookDetails	
bookid	int
title	string
author	string
pages	int
edition	int
release	int
genre	string
image	string
format	string
special	string
description	string
metamask	string
marketprice	int
userprice	int
upforsale	boolean

Smart Contract Deployment on Sepolia Network via Infura

Sign up on Infura to create an account. Log in to view dashboard. Choose Sepolia under Ethereum networks to get url with api key for smart contract deployment.



Load the smart contract files under BookChain-Contract folder. Use npm install @truffle/hdwallet-provider command to install hd-wallet provider.

```
(base) ashwin@Ashwins-MBP BookChain-Contract % npm install @truffle/hdwallet-provider
```

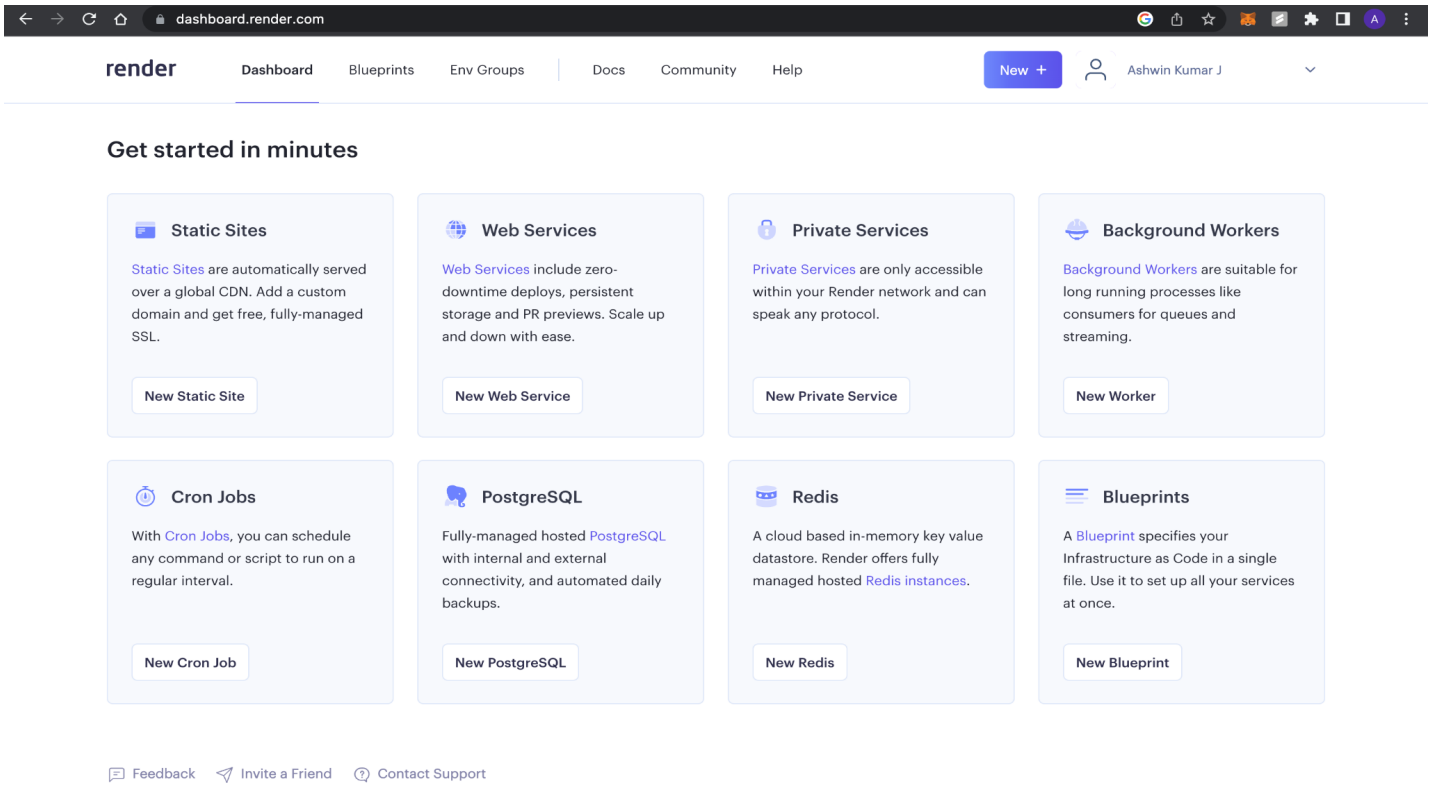
Update the truffle-config js file to make sure contracts are deployed on sepolia network. Now move to BookChain-Contract folder to compile the smart contracts. Then use truffle migrate --reset --network sepolia to deploy the smart contracts on sepolia network. Once deployed, json files are created for every smart contract.

```
(base) ashwin@Ashwins-MBP BookChain-Contract % truffle migrate --reset --network sepolia

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


Migrations dry-run (simulation)
===============================
> Network name:    'sepolia-fork'
> Network id:      11155111
> Block gas limit: 30000000 (0x1c9c380)


2_deploy_contracts.js
=====================

   Replacing 'Bookies'
   -------------------
   > block number:        3447952
   > block timestamp:     1683606776
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.495518992496512755
   > gas used:            1237979 (0x12e3db)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.003094947508665853 ETH


   Replacing 'BookNFT'
   -------------------
   > block number:        3447953
   > block timestamp:     1683606778
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.490074389981267868
   > gas used:            2177841 (0x213b31)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.005444602515244887 ETH


   Replacing 'Bookchain'
   ---------------------
   > block number:        3447954
   > block timestamp:     1683606780
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.485208874967644426
   > gas used:            1946206 (0x1db25e)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.004865515013623442 ETH

   ---------------------------------------
   > Total cost:     0.013405065037534182 ETH

Summary
=======
> Total deployments:   3
> Final cost:          0.013405065037534182 ETH
```

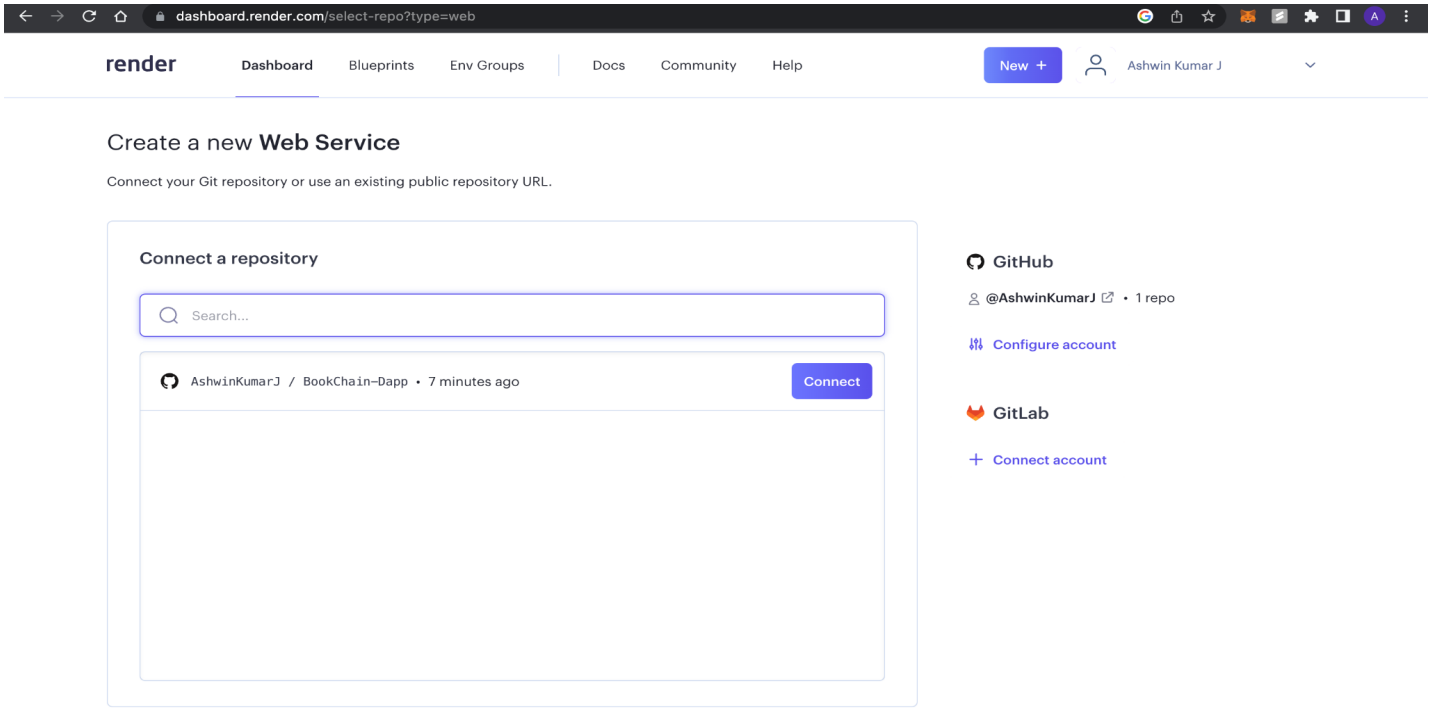
```
> Network name:    'sepolia'
> Network id:      11155111
> Block gas limit: 30000000 (0x1c9c380)


2_deploy_contracts.js
=====================

   Replacing 'Bookies'
   -------------------
   > transaction hash:    0x378abdadb8ed939067a19cde3724303e432a6df8673ffca0ad6f2e9f8e0b8ad5
   > Blocks: 0            Seconds: 16
   > contract address:    0x30d9B6A6d2f762A558209A2c6cd9b4f902778e2f
   > block number:        3447957
   > block timestamp:     1683606804
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.495518007496509997
   > gas used:            1238373 (0x12e565)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.003095932508668611 ETH


   Replacing 'BookNFT'
   -------------------
   > transaction hash:    0x80cdac4337e4e756031bf7ae5940bb9d2d7517fc4b6298b6e5a14b1eb612f6d0
   > Blocks: 0            Seconds: 4
   > contract address:    0x18e9404526dB7D7c411a7Ddc09349019c31C0A58
   > block number:        3447958
   > block timestamp:     1683606816
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.490071734981260434
   > gas used:            2178509 (0x213dcd)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.005446272515249563 ETH


   Replacing 'Bookchain'
   ---------------------
   > transaction hash:    0x89223a42ce3b2a40f96616b58e176ce517ca72731a5806d08355e559a429a8e2
   > Blocks: 1            Seconds: 8
   > contract address:    0xAe34ee0891db965840Bf70e33fefFF13ead22E2c
   > block number:        3447959
   > block timestamp:     1683606828
   > account:             0xF70Be17087FaA85Dac9Fb3831516966285eadda5
   > balance:             0.485204819967633072
   > gas used:            1946766 (0x1db48e)
   > gas price:           2.500000007 gwei
   > value sent:          0 ETH
   > total cost:          0.004866915013627362 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:     0.013409120037545536 ETH

Summary
=======
> Total deployments:   3
> Final cost:          0.013409120037545536 ETH


(base) ashwin@Ashwins-MBP BookChain-Contract %
```

Application Hosting via Render

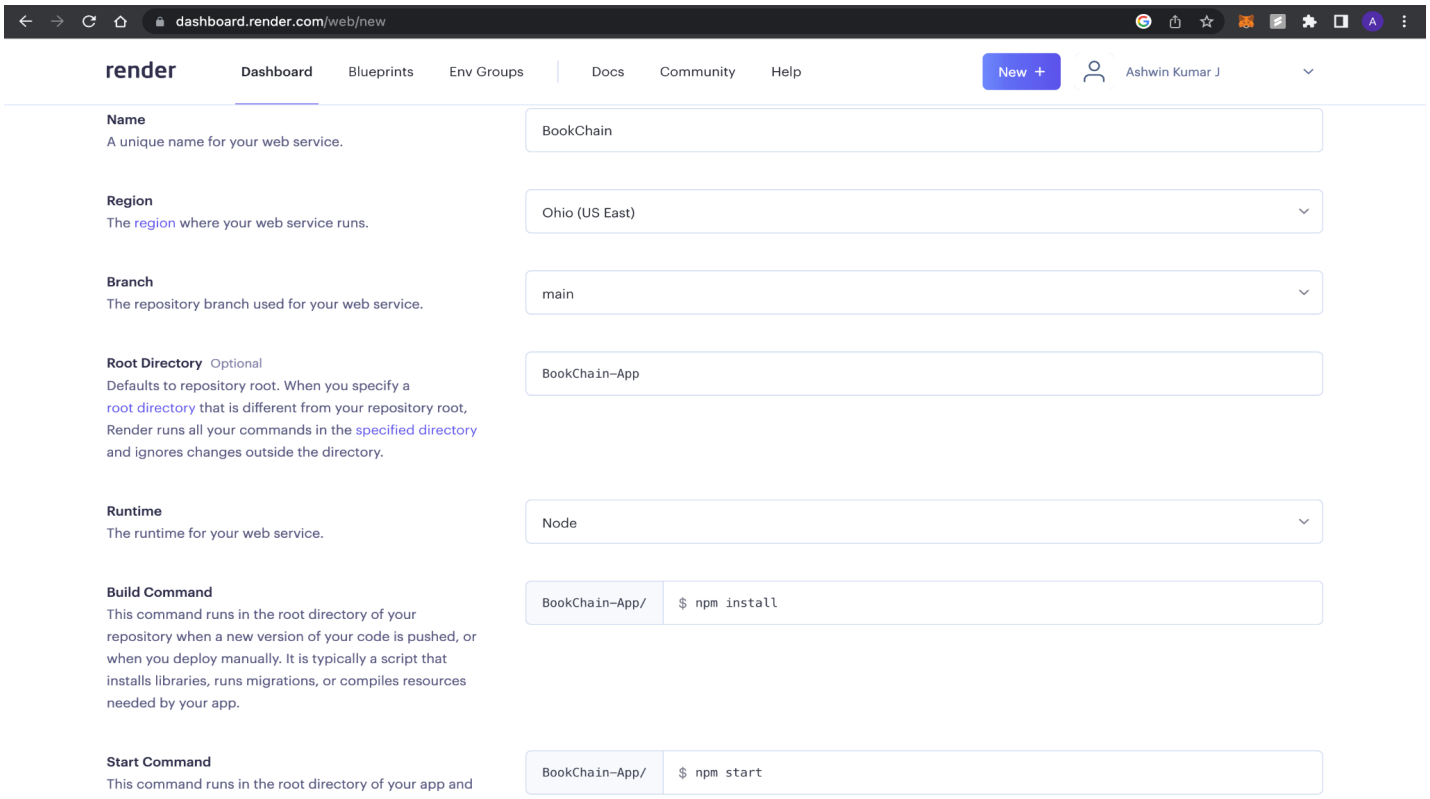
Sign up on Render. Choose new web service to host our application.



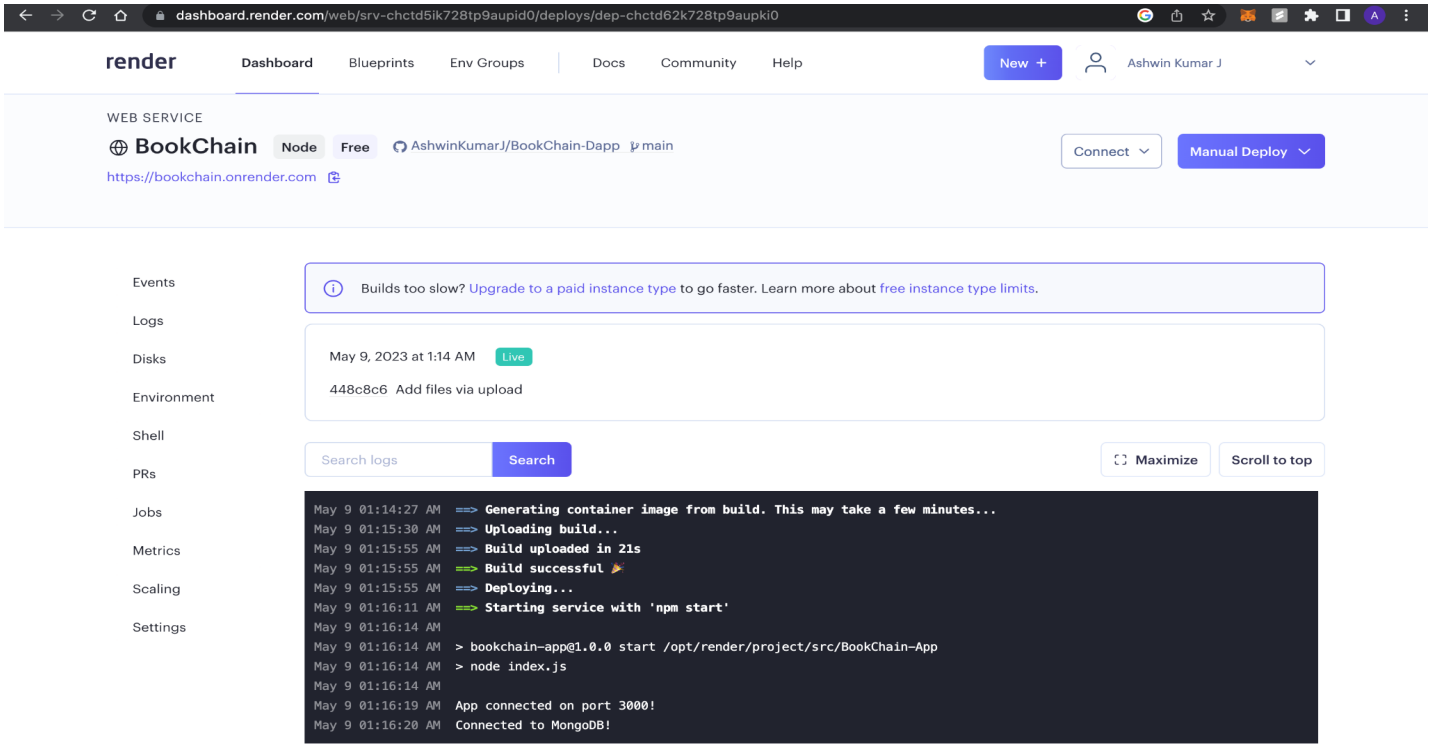
Now a list of all repositories in your Github account is shown. Connect to Github repository with the BookChain application code.



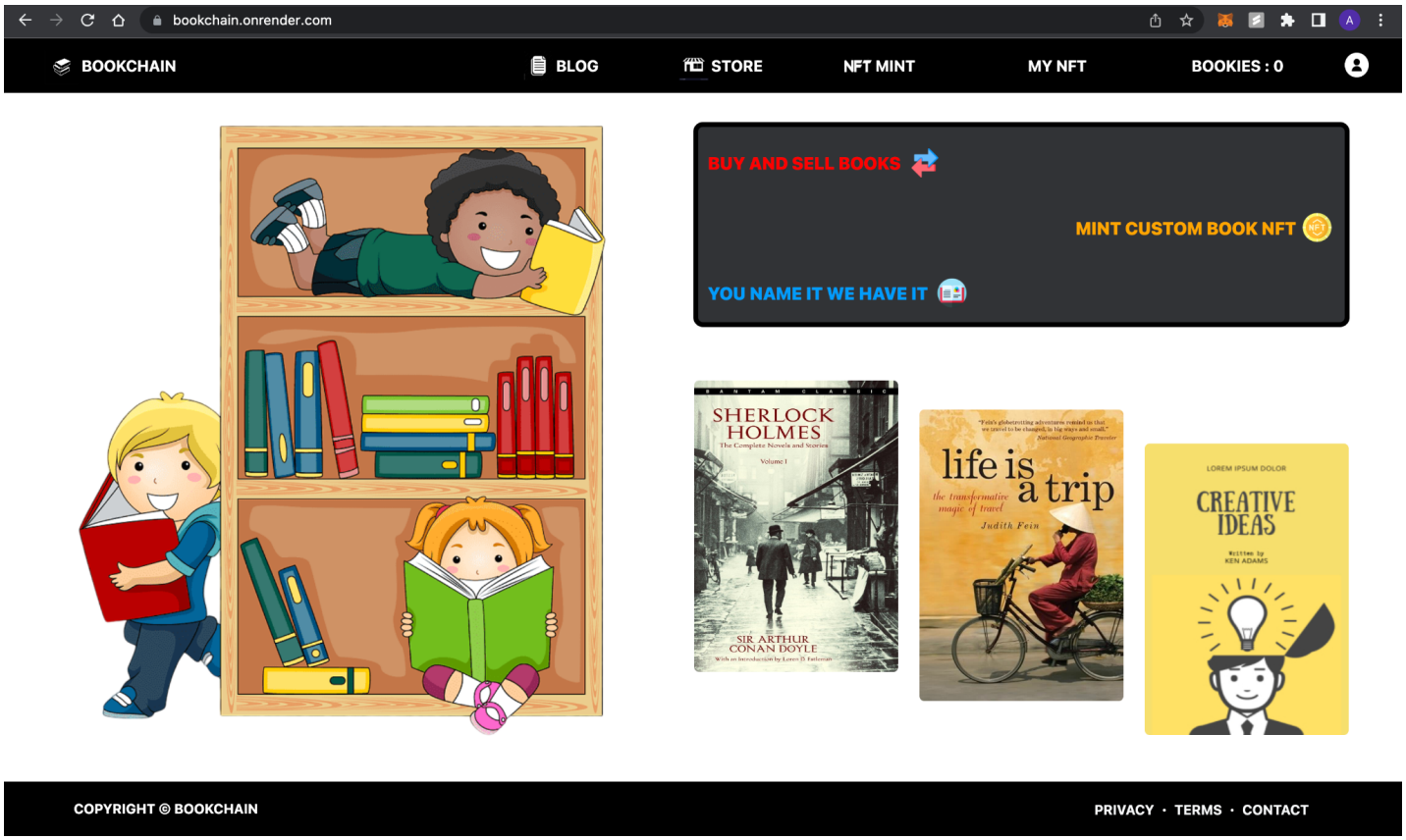
Enter the details of the BookChain application and specify the root directory and the commands for installation, deployment and launching the application.



Once the web service is created, the deployment starts using the commands provided. After deployment, the application is hosted and connected to MongoDB.



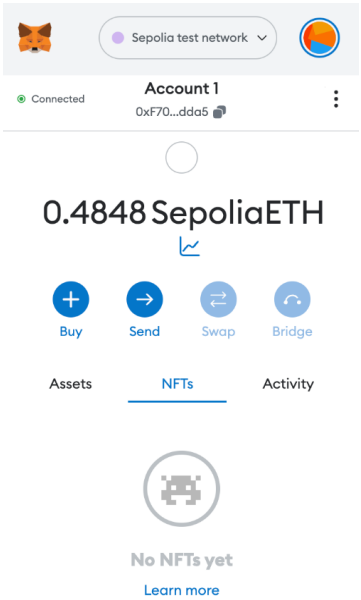
When we hit the url provided for the hosted application : https://bookchain.onrender.com , the Home page is displayed.



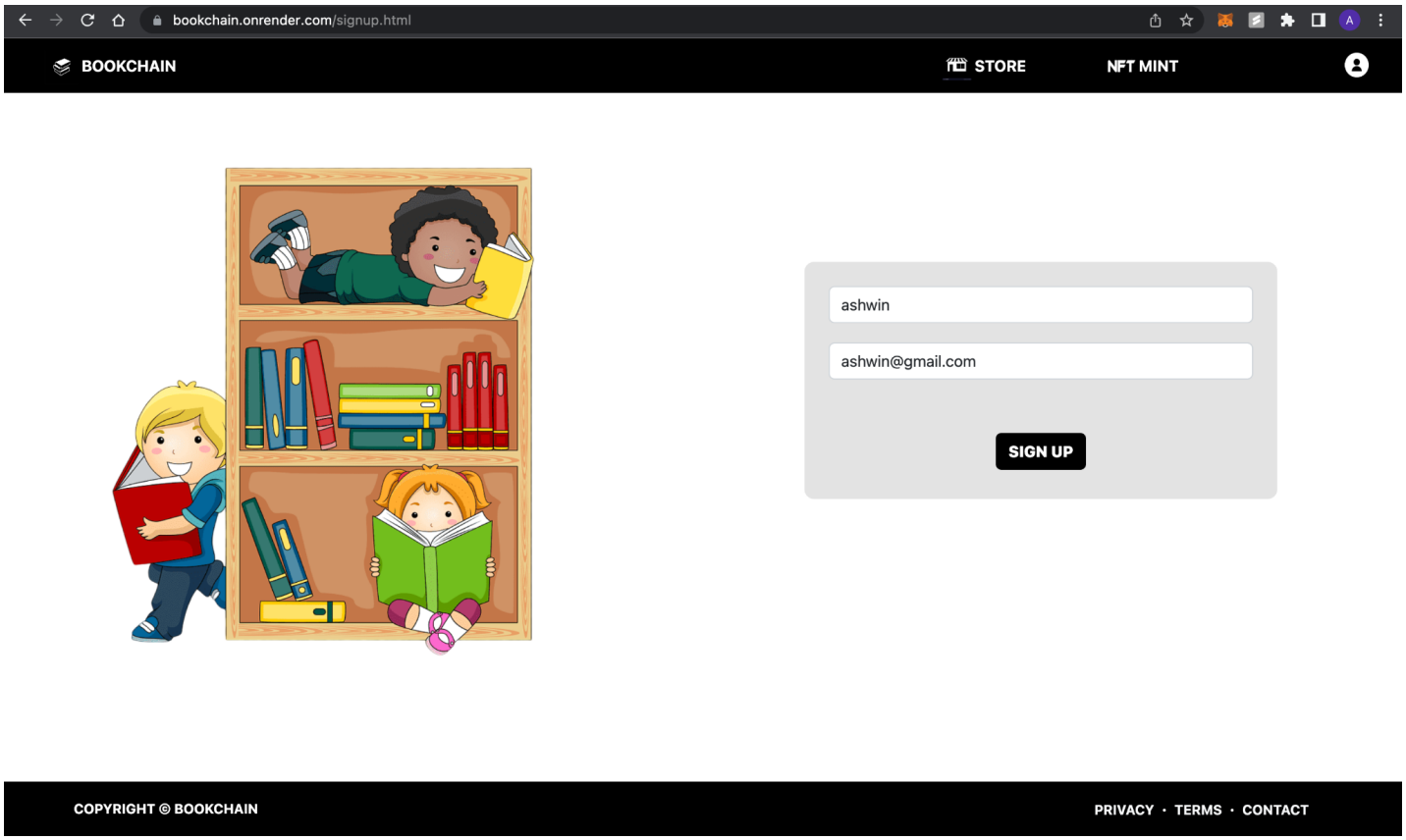
Application Workflow

Sign up a new user to our BookChain application. The user needs to sign up using his metamask account. The user chooses his account address from metamask.

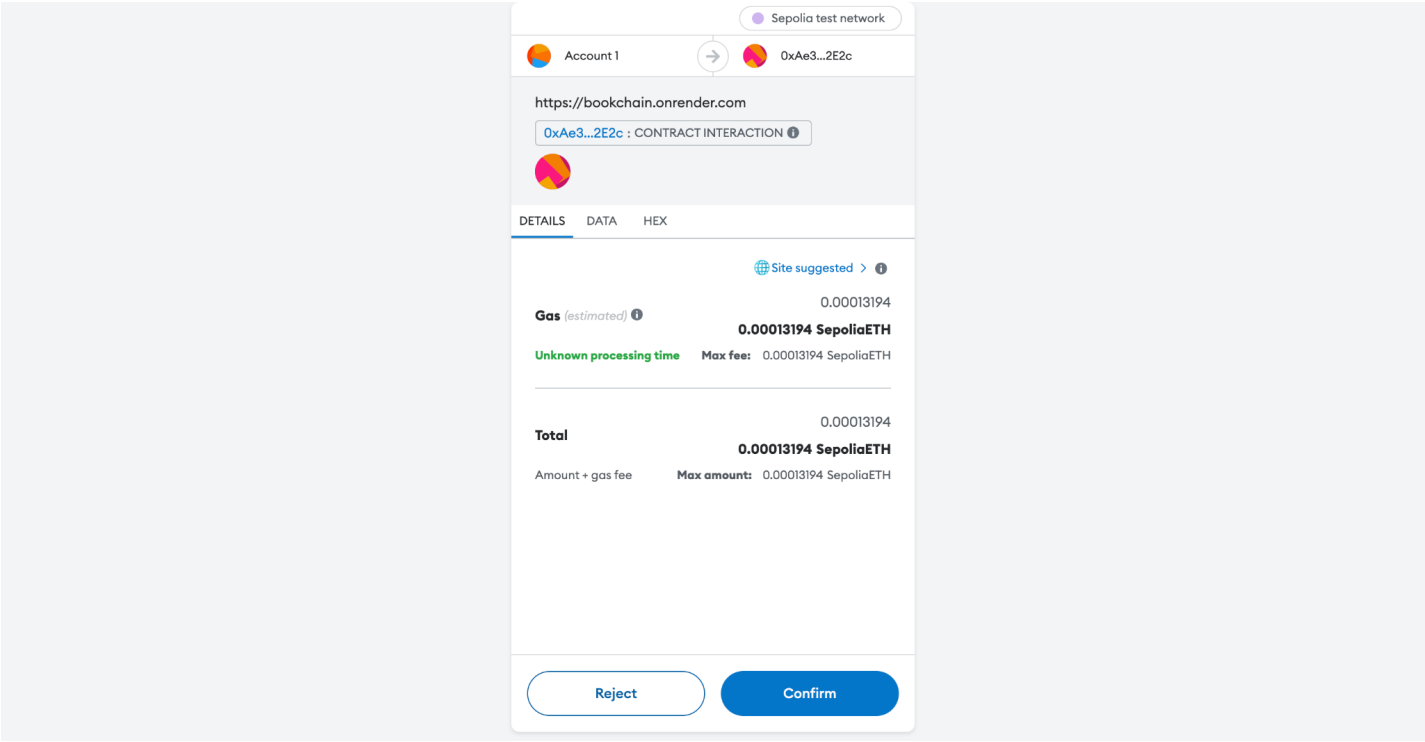
User 1 : 0xF70Be17087FaA85Dac9Fb3831516966285eadda5



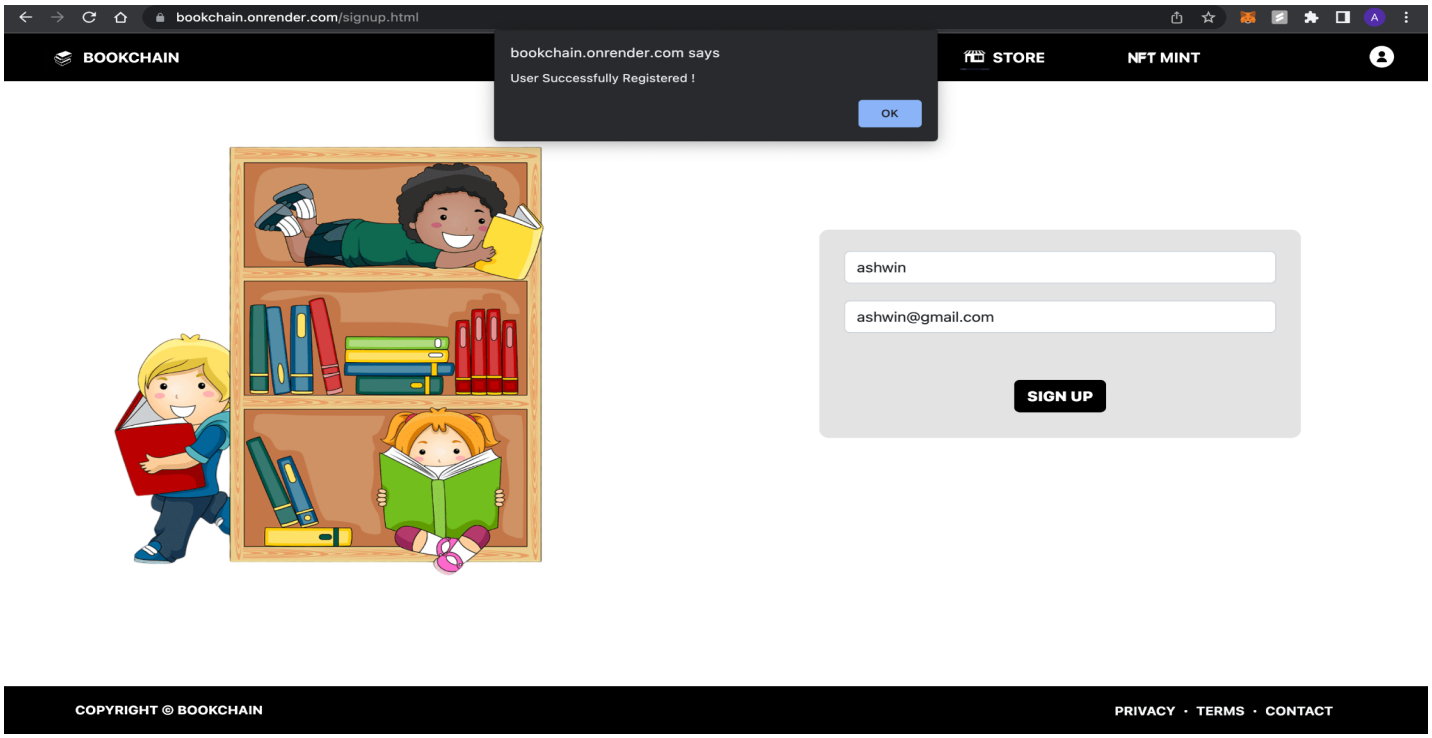
Initially the user has 0 Bookies in his account. The user signs up with a username and email address.



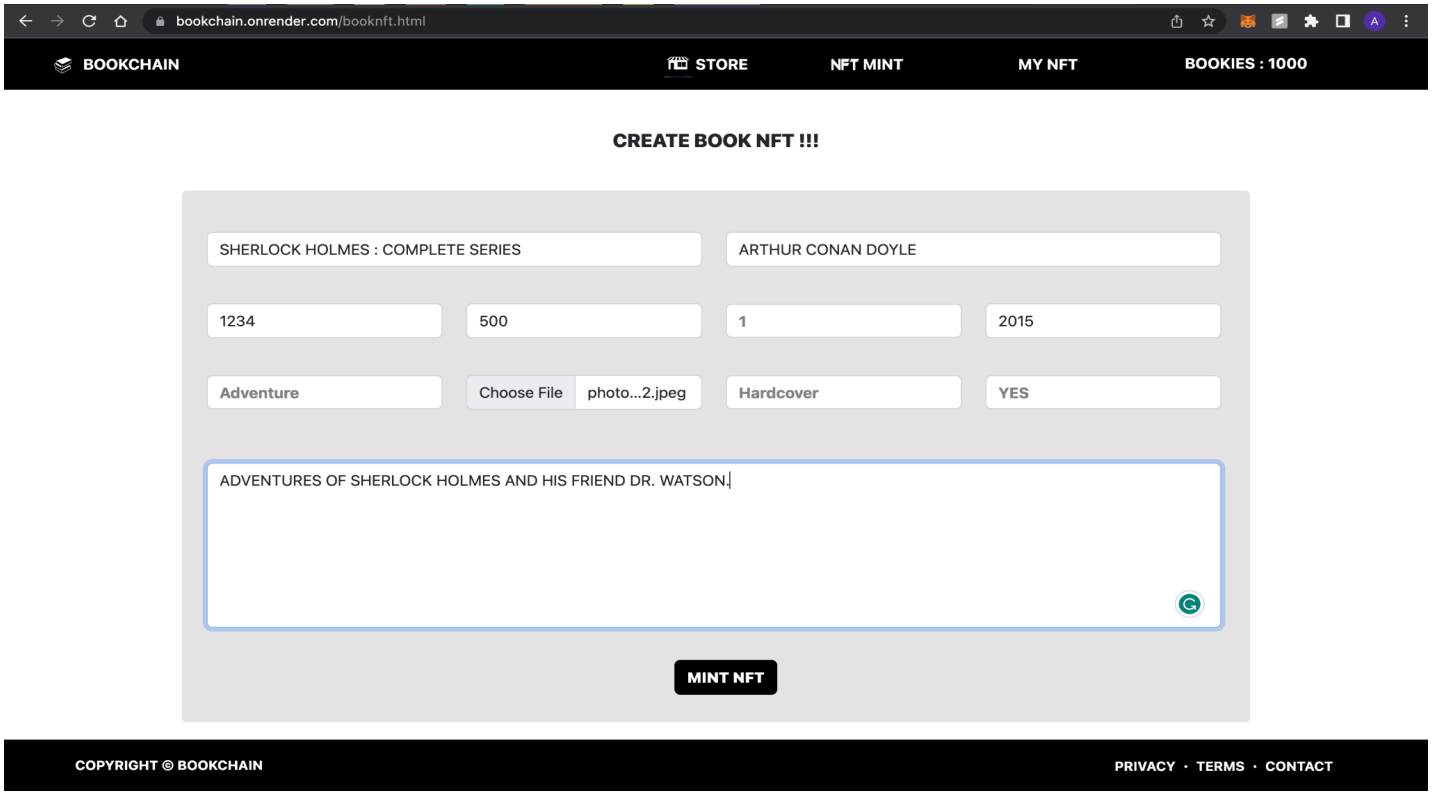
Once we hit sign up, the application redirects to the metamask page to confirm the transaction.



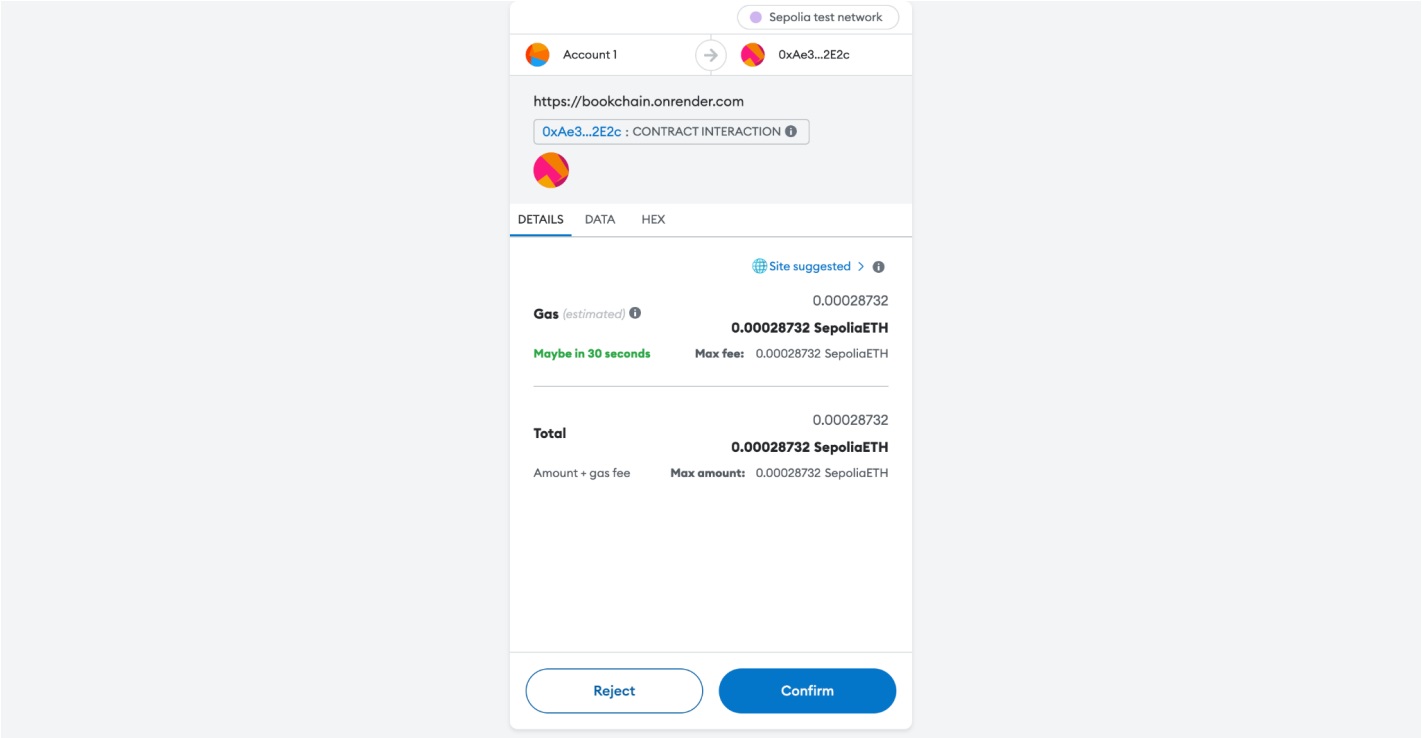
Once the transaction is confirmed, the user is notified by an alert from the application on successful signing up.



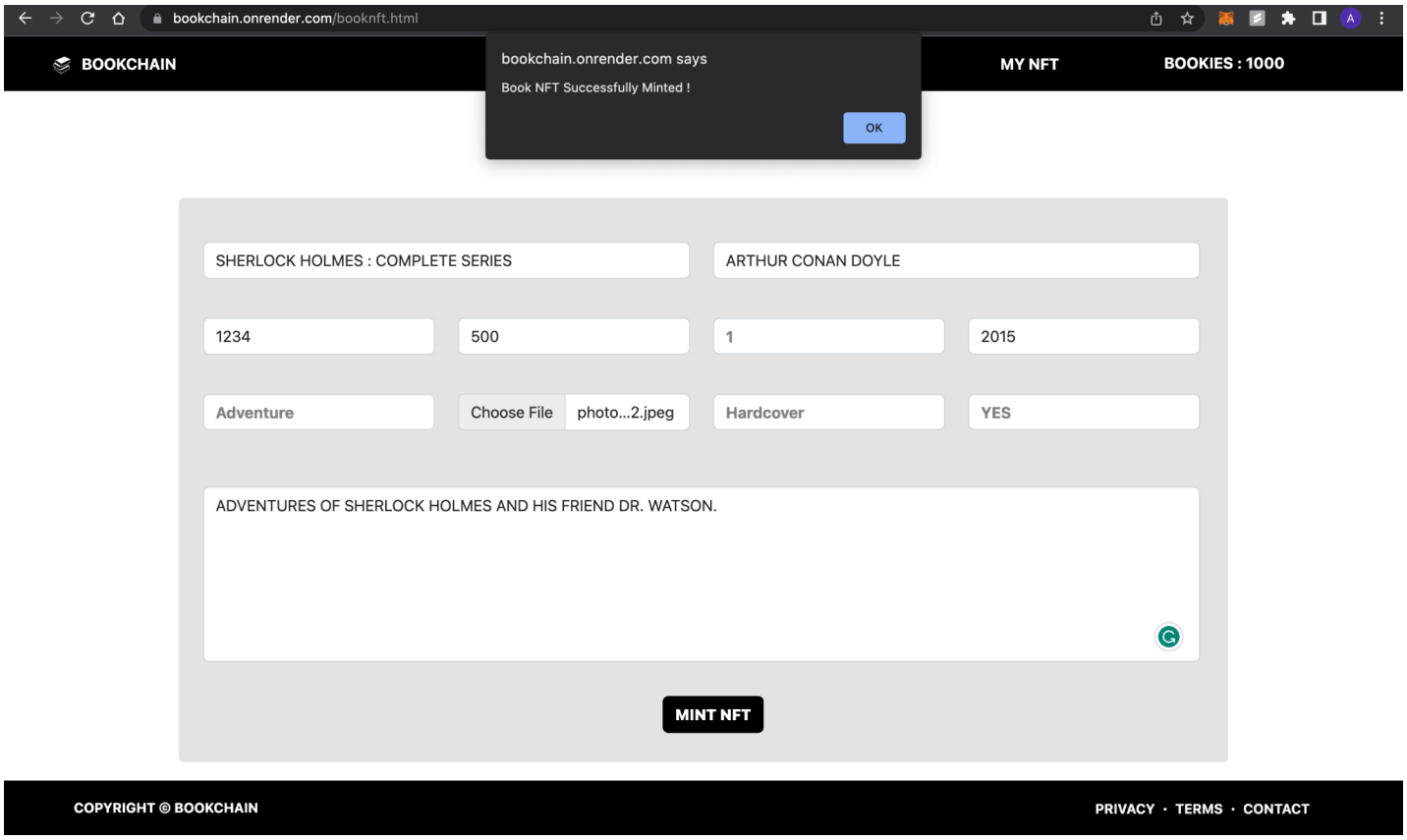
Once a user is signed up, he is provided with initial 1000 Bookies. Now the user can create a Book NFT for his own collection with required book details. Each book has a bookid that uniquely identifies a Book NFT and used in minting the NFT.



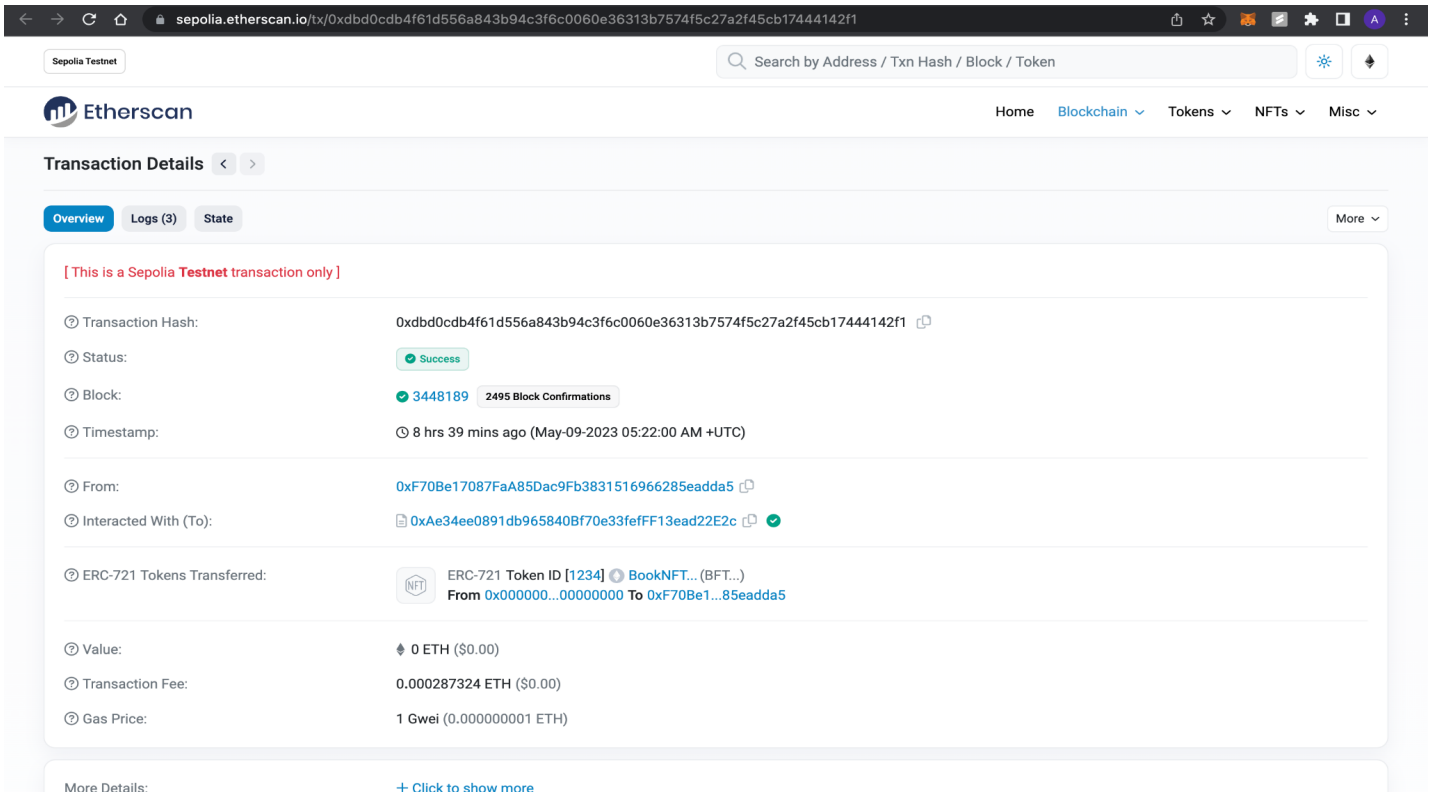
Once the user hits mint nft, the application redirects to the metamask page to confirm the transaction.



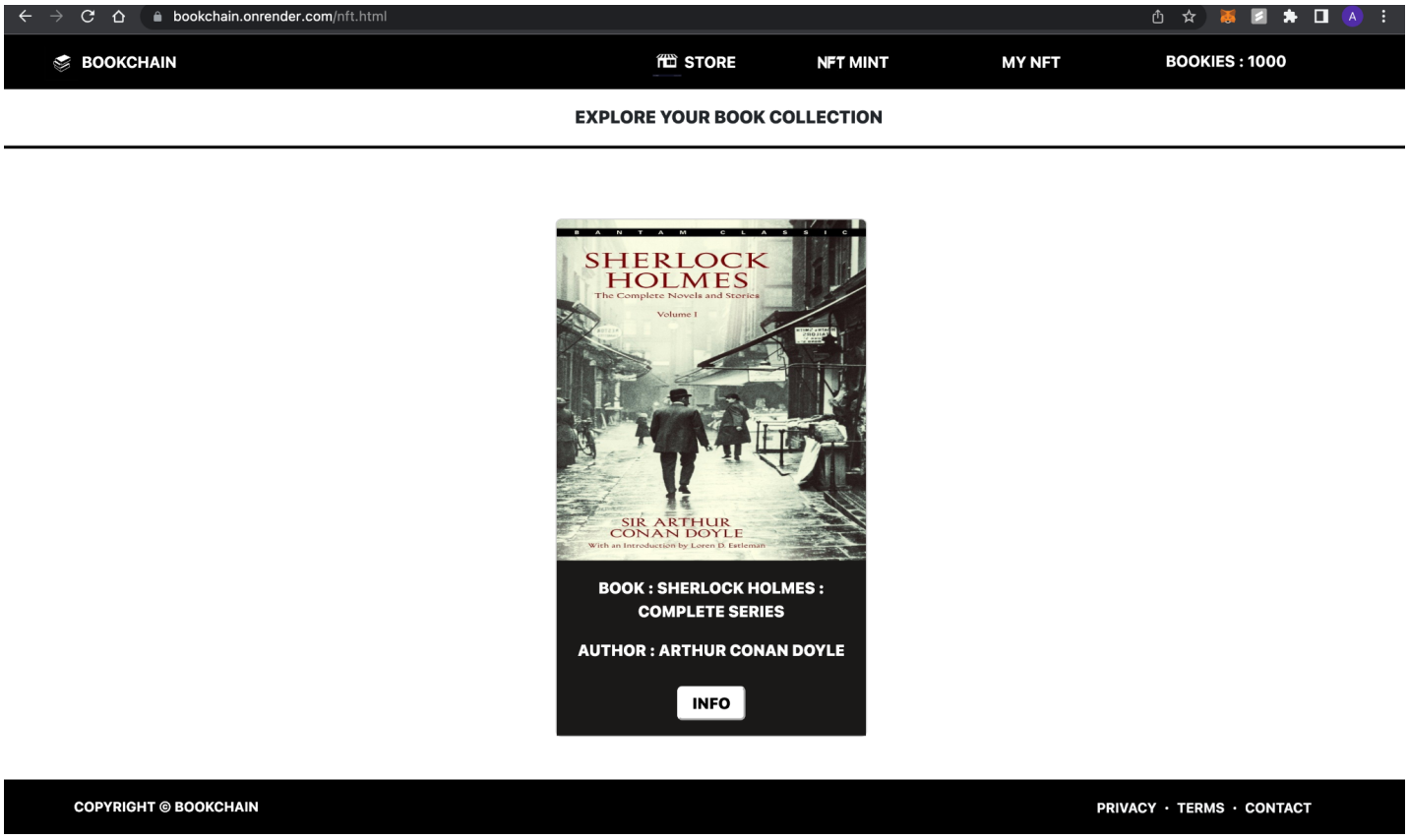
Once the transaction is confirmed, a Book NFT is created. The user is also notified via an alert from the application.



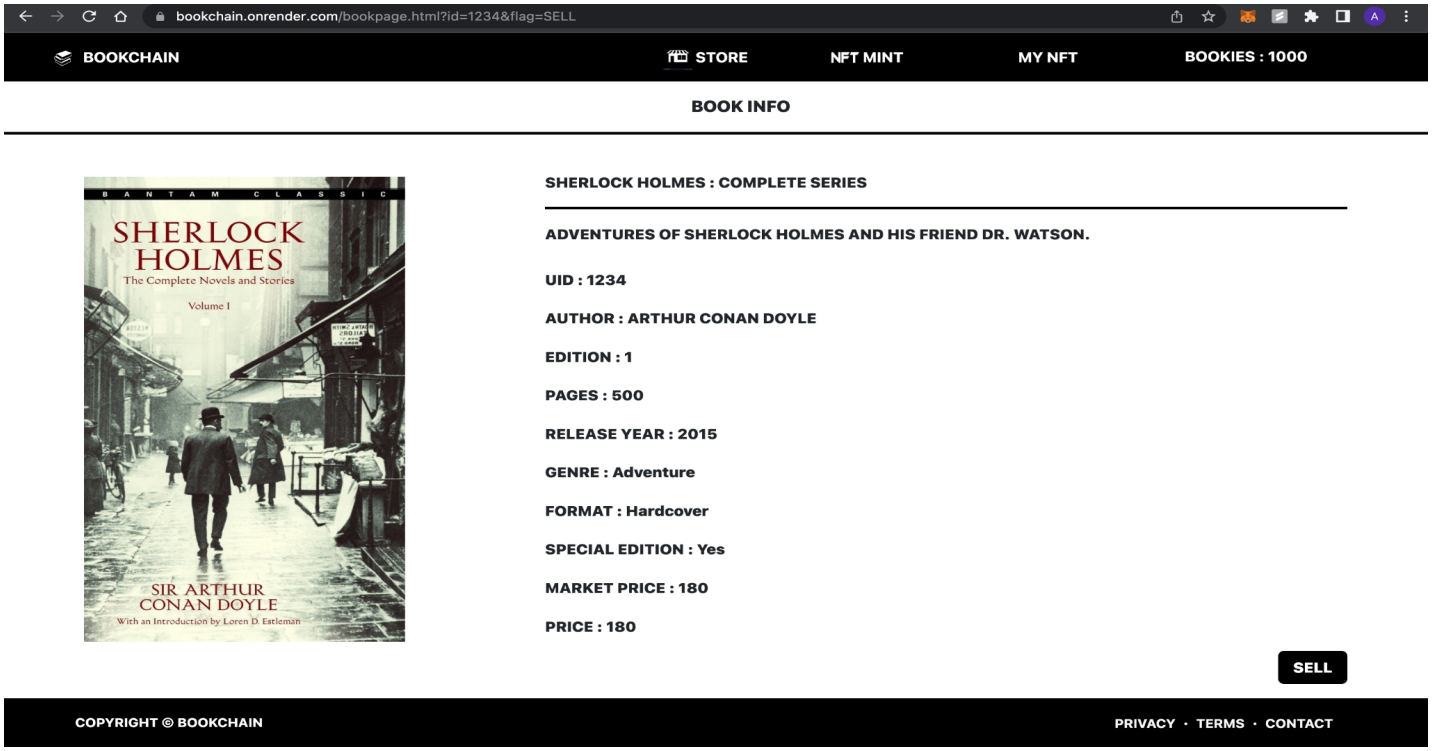
The Book NFT minted can also be verified from the Etherscan logs.



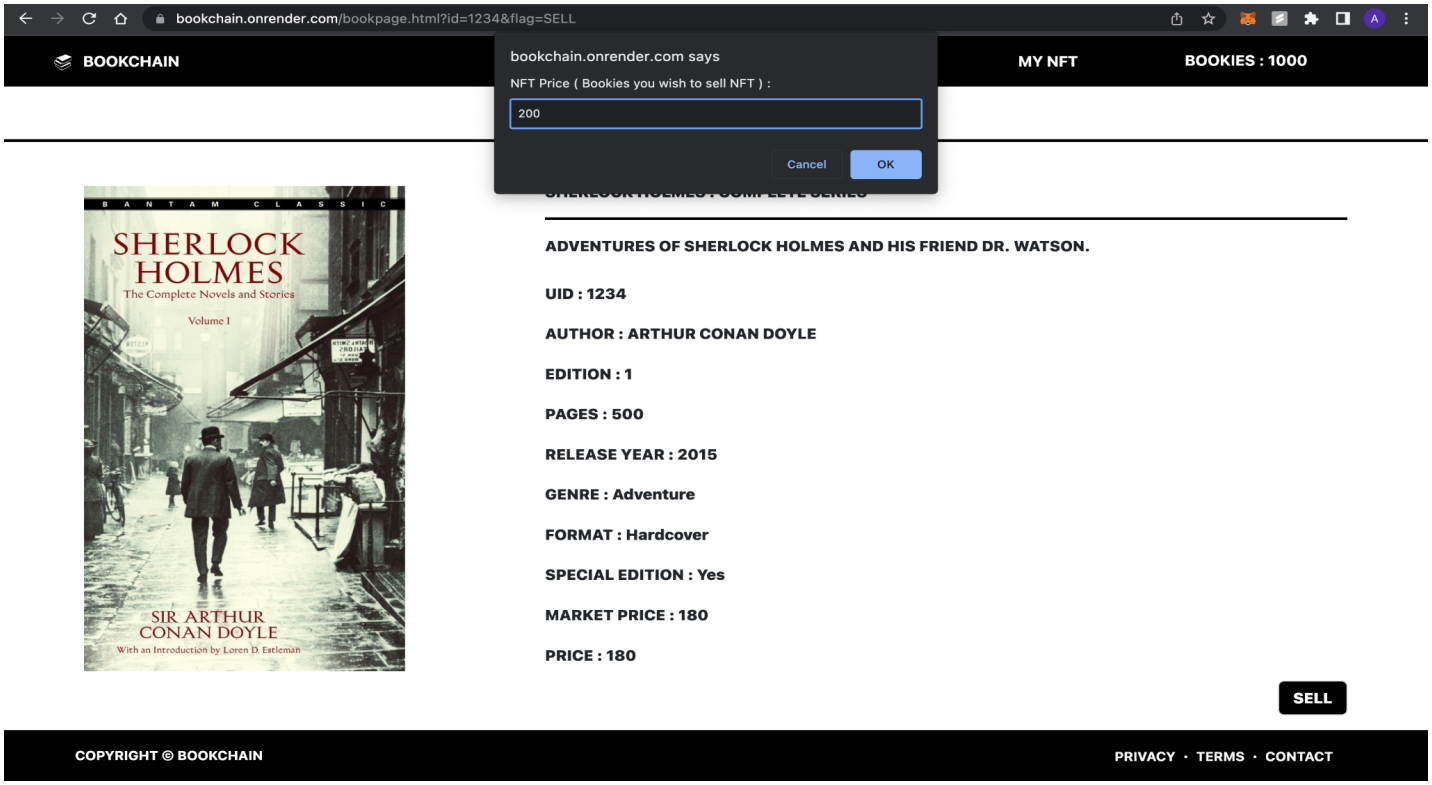
The Book NFT is added to the user 1 inventory page.



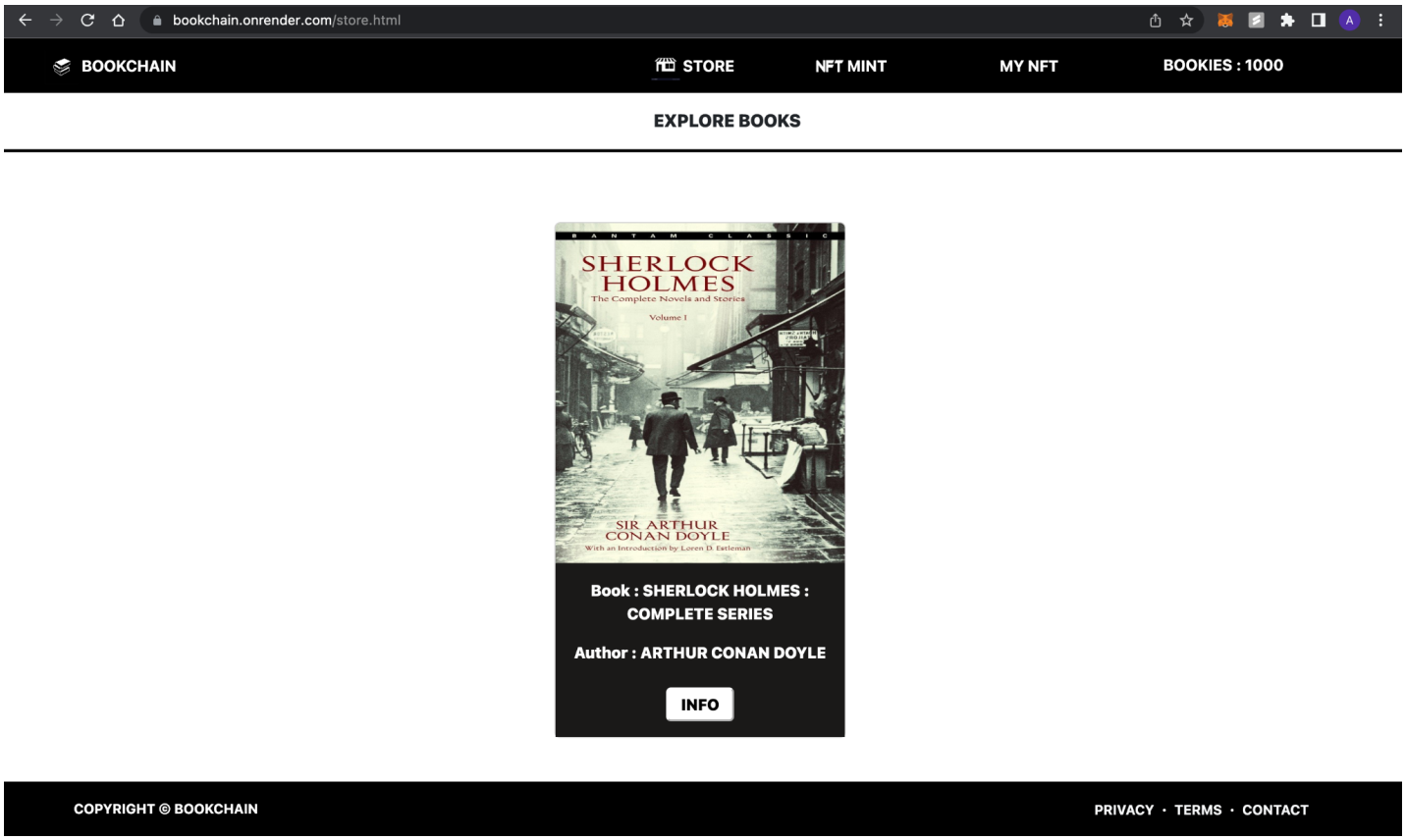
The user can view the details of his Book NFT once he hits the info button. The Book Info page displays each and every detail of the book along with the market price of the Book NFT minted, depending on various factors : **Edition, Release Year, Genre, Format & Special Edition**.



Once the user decides to put his NFT up for sale, he can hit the sell button. This triggers a prompt asking the value he would like to sell his NFT. Once the user enters the number of Bookies and hits ok, the Book NFT is up for sale.

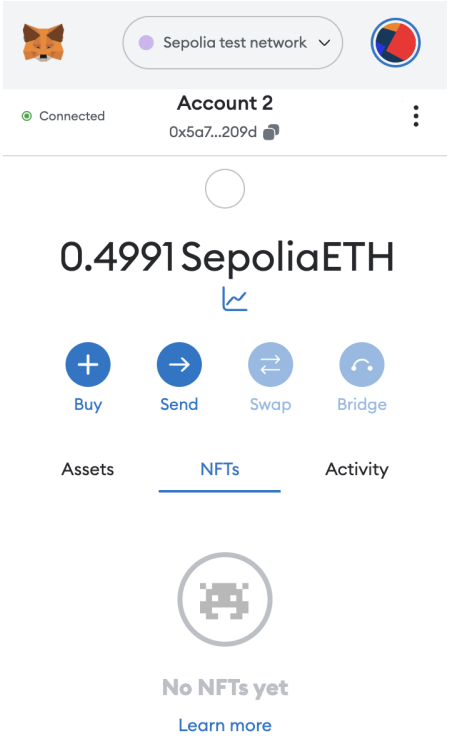


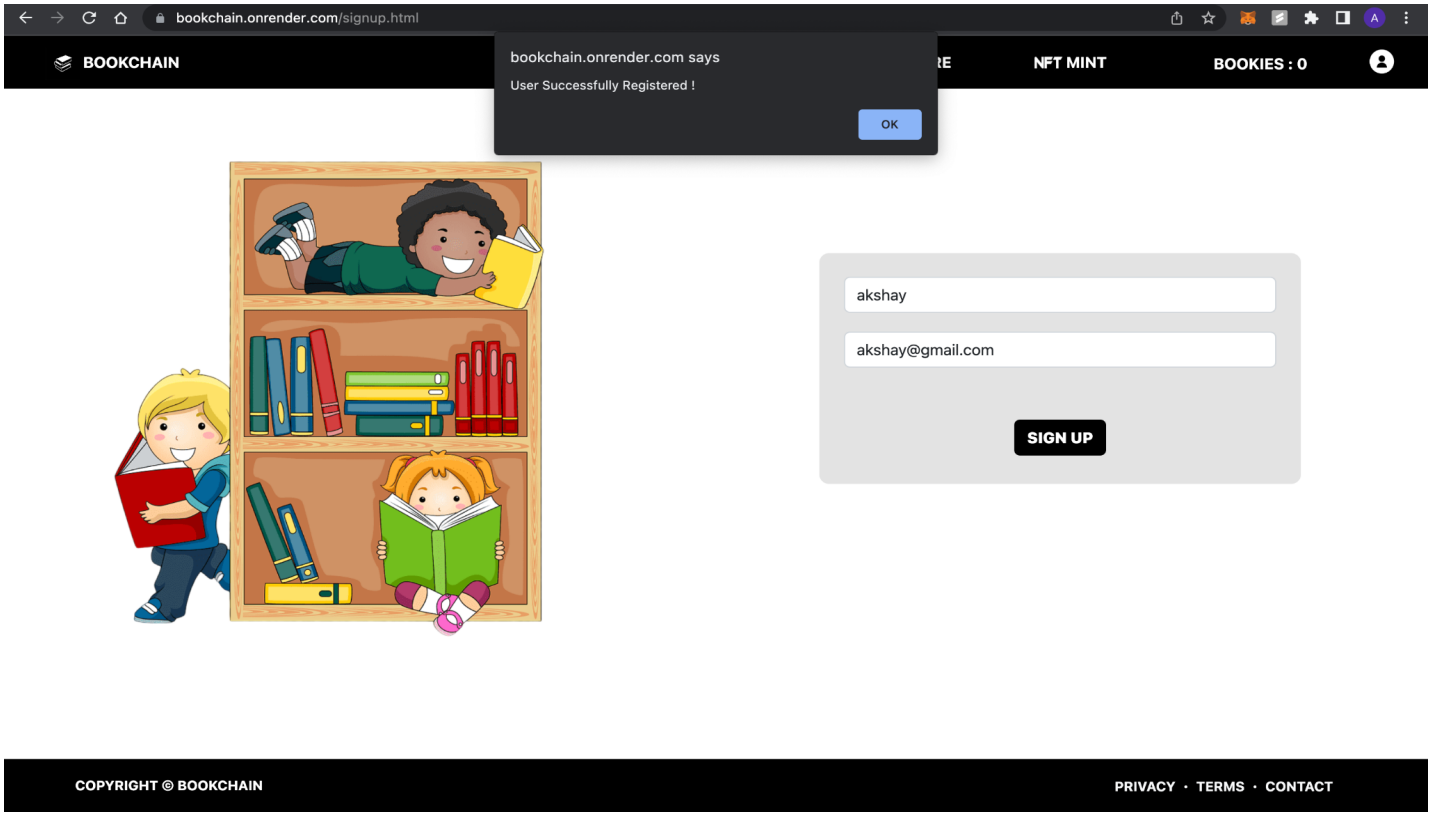
Now the user can see his NFT available on the store for purchase.



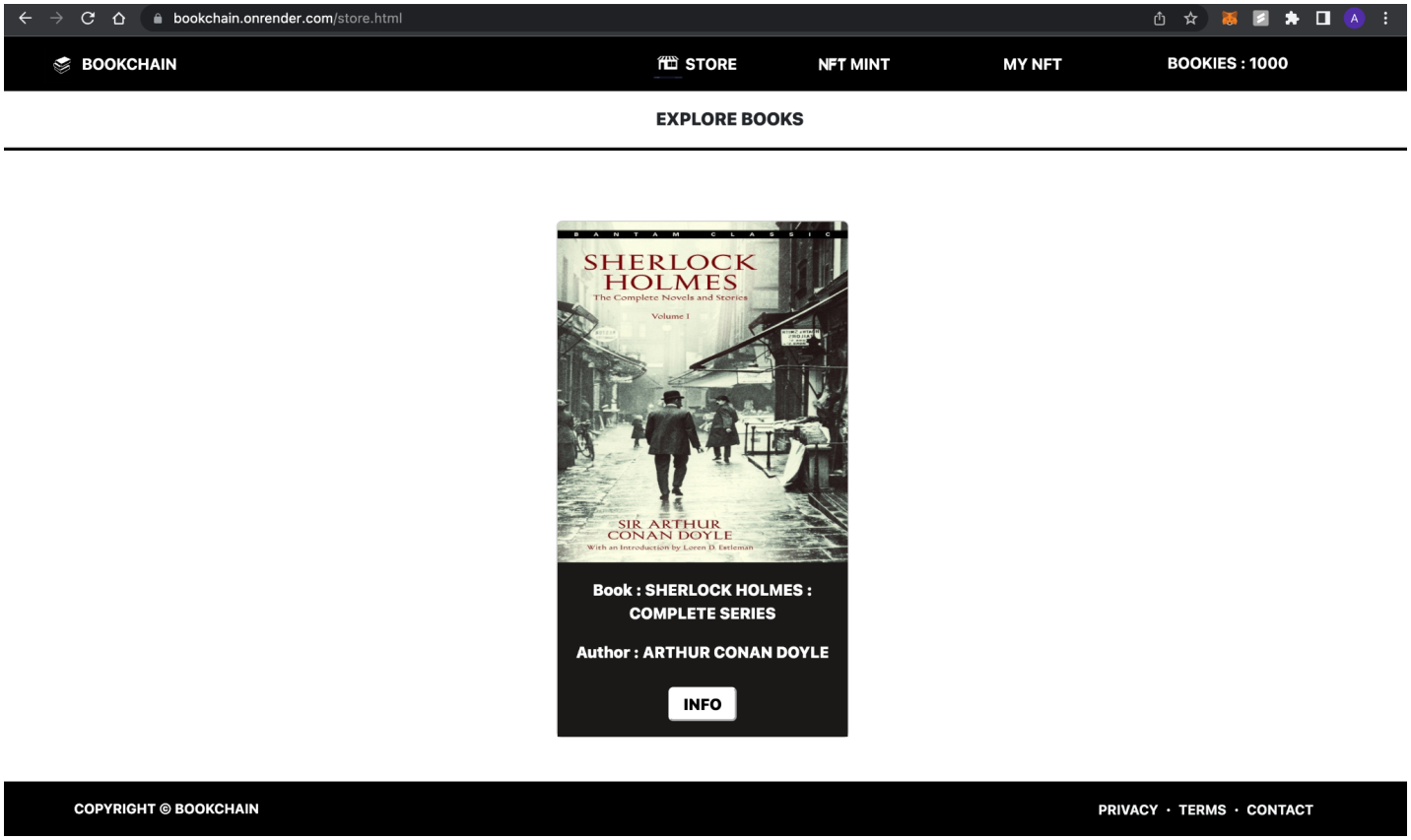
Now we sign up a new user with different account address to be able to buy the Book NFT minted. The user is also provided with initial 1000 Bookies.

User 2 : 0x5a70cD0b2e7AF7bE826A8A235AaD7f2995aa209d

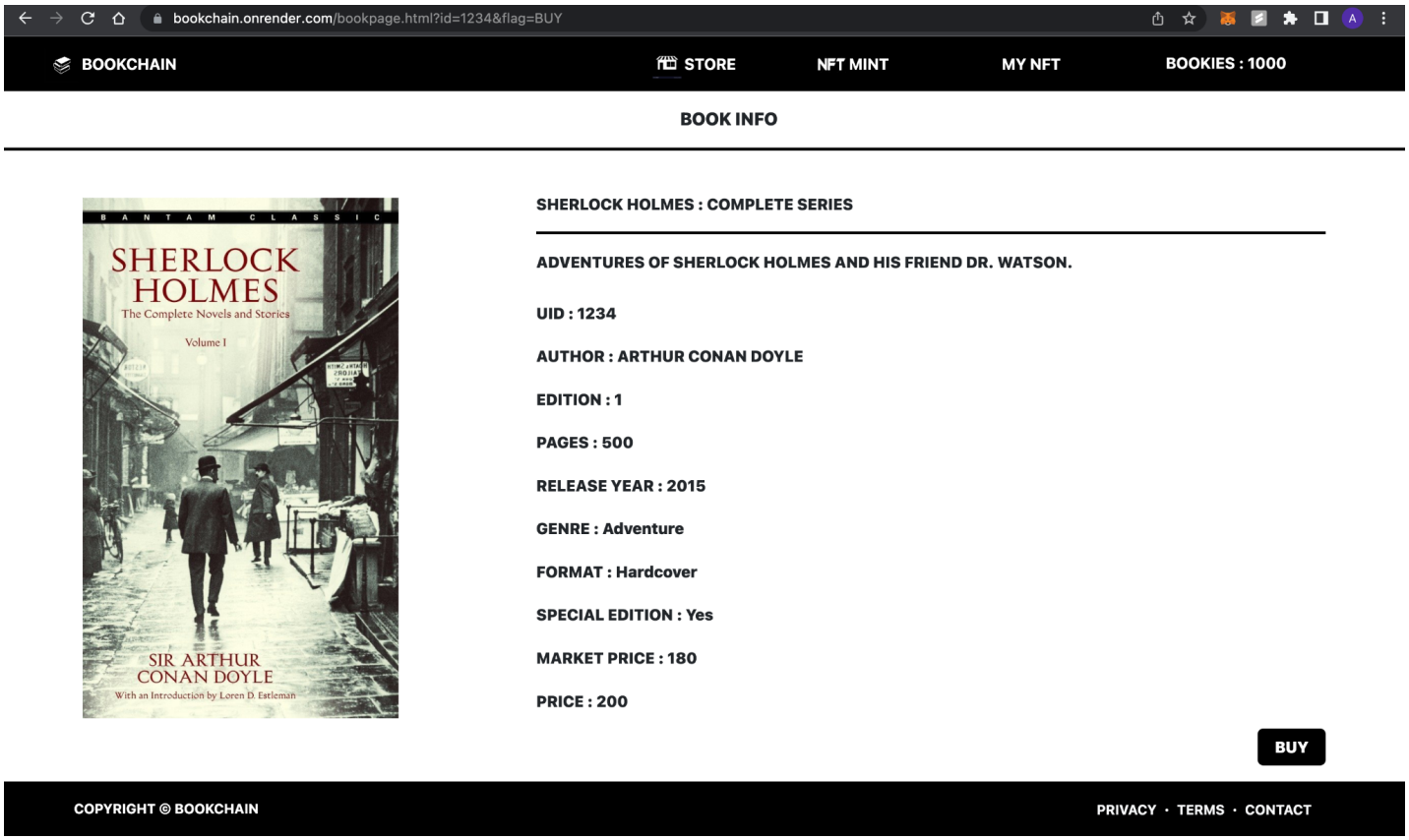




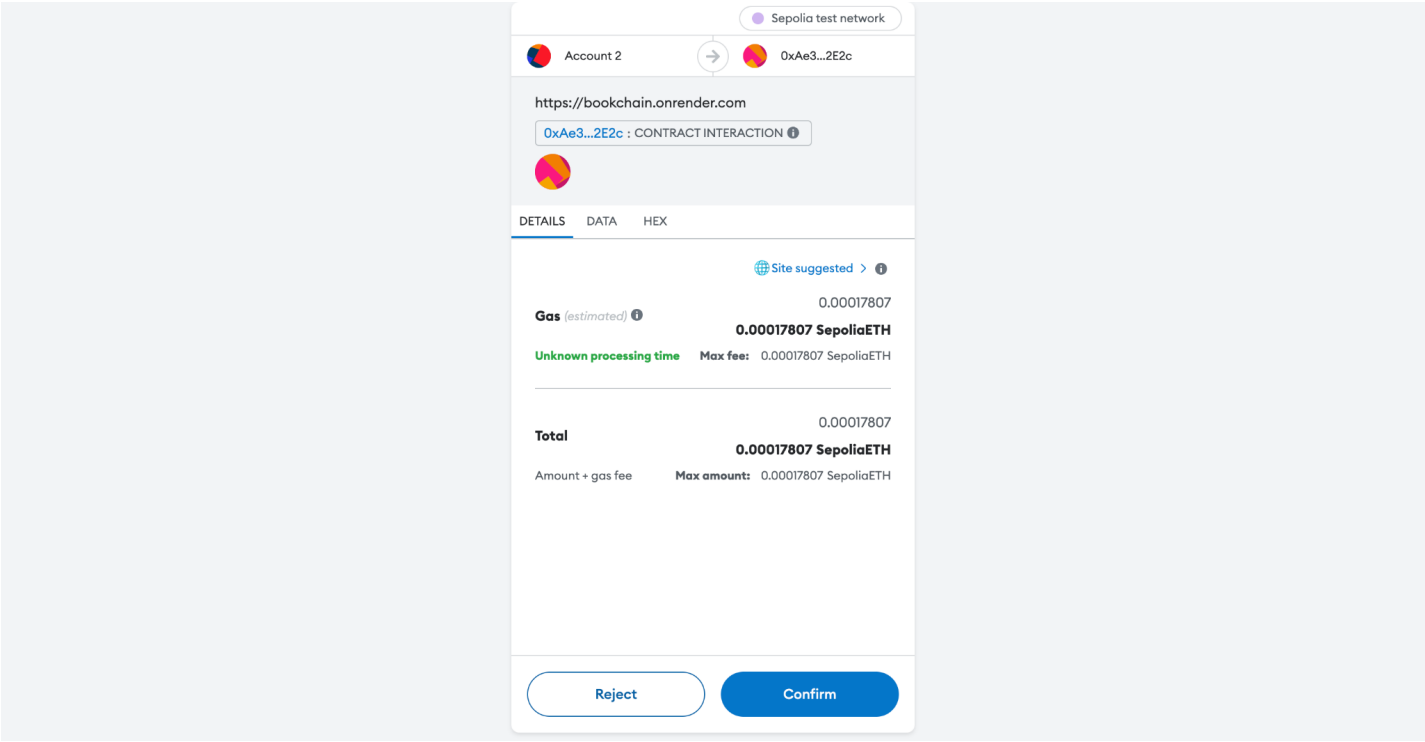
Now the user's inventory is empty since he has not minted an NFT for his own collection. He can view the Book NFTs available for purchase on the store.



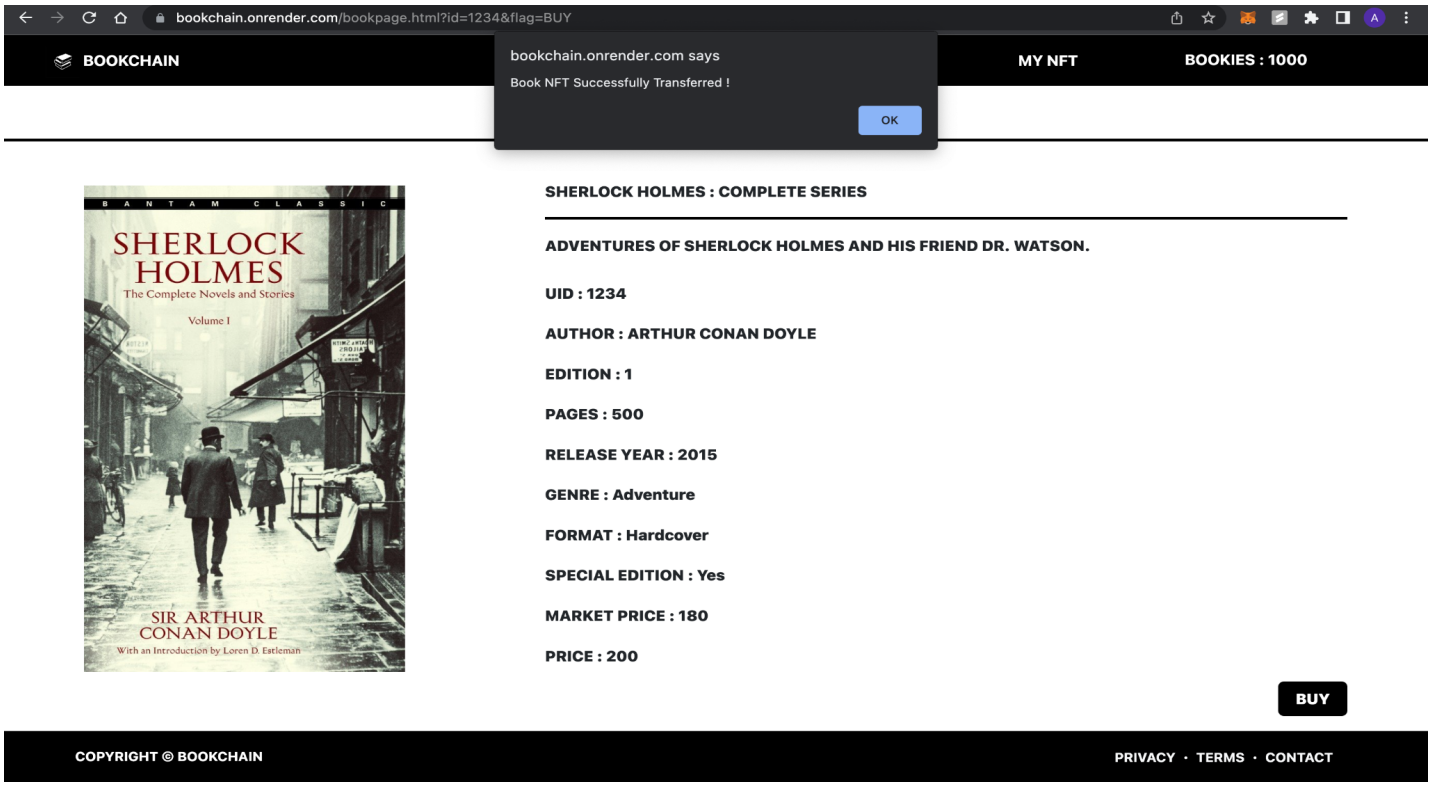
User can view the details of the Book NFT, including the price the user has set for the purchase of NFT.



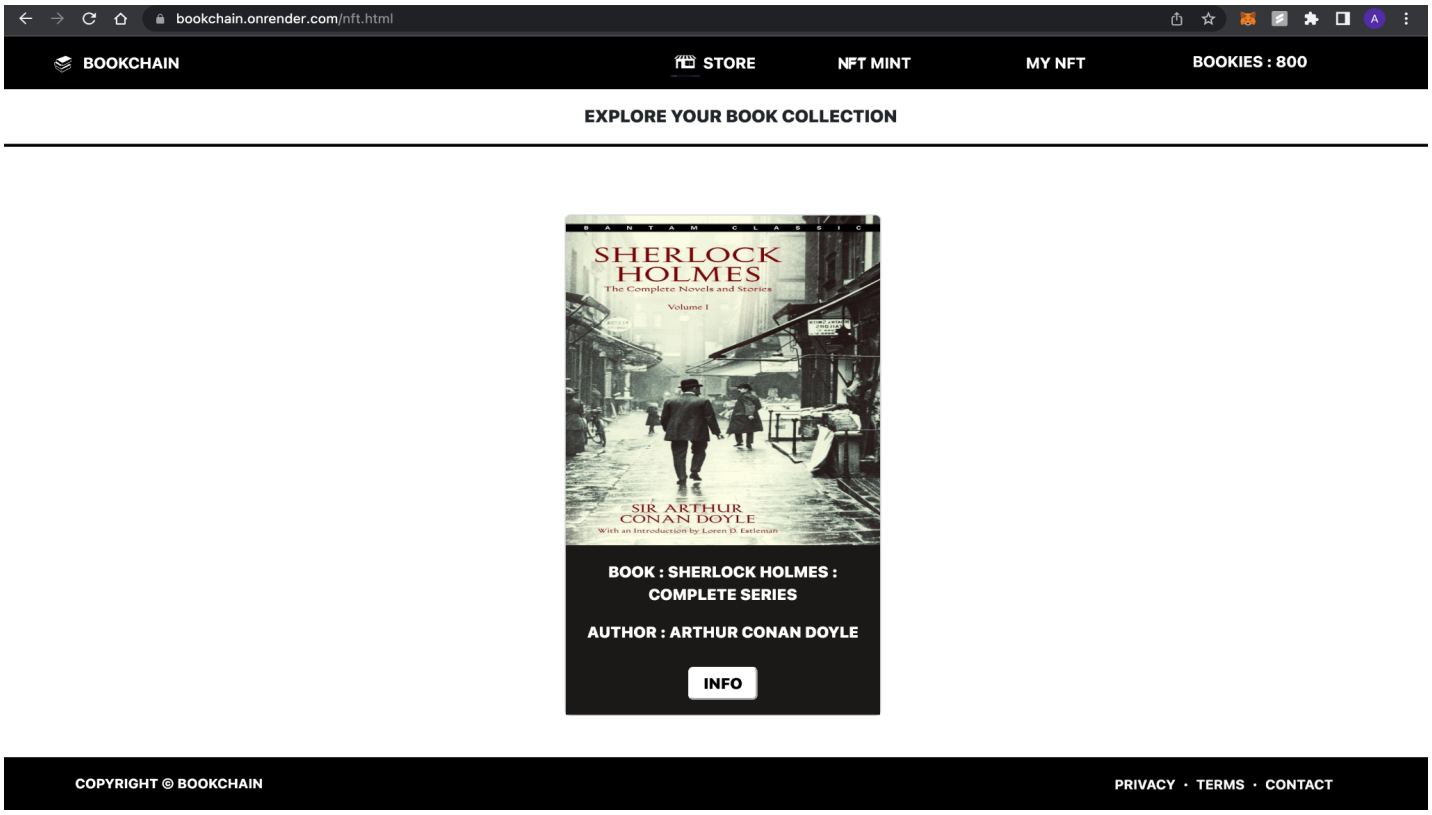
Once he hits buy, the application redirects to the metamask page to confirm the transaction.



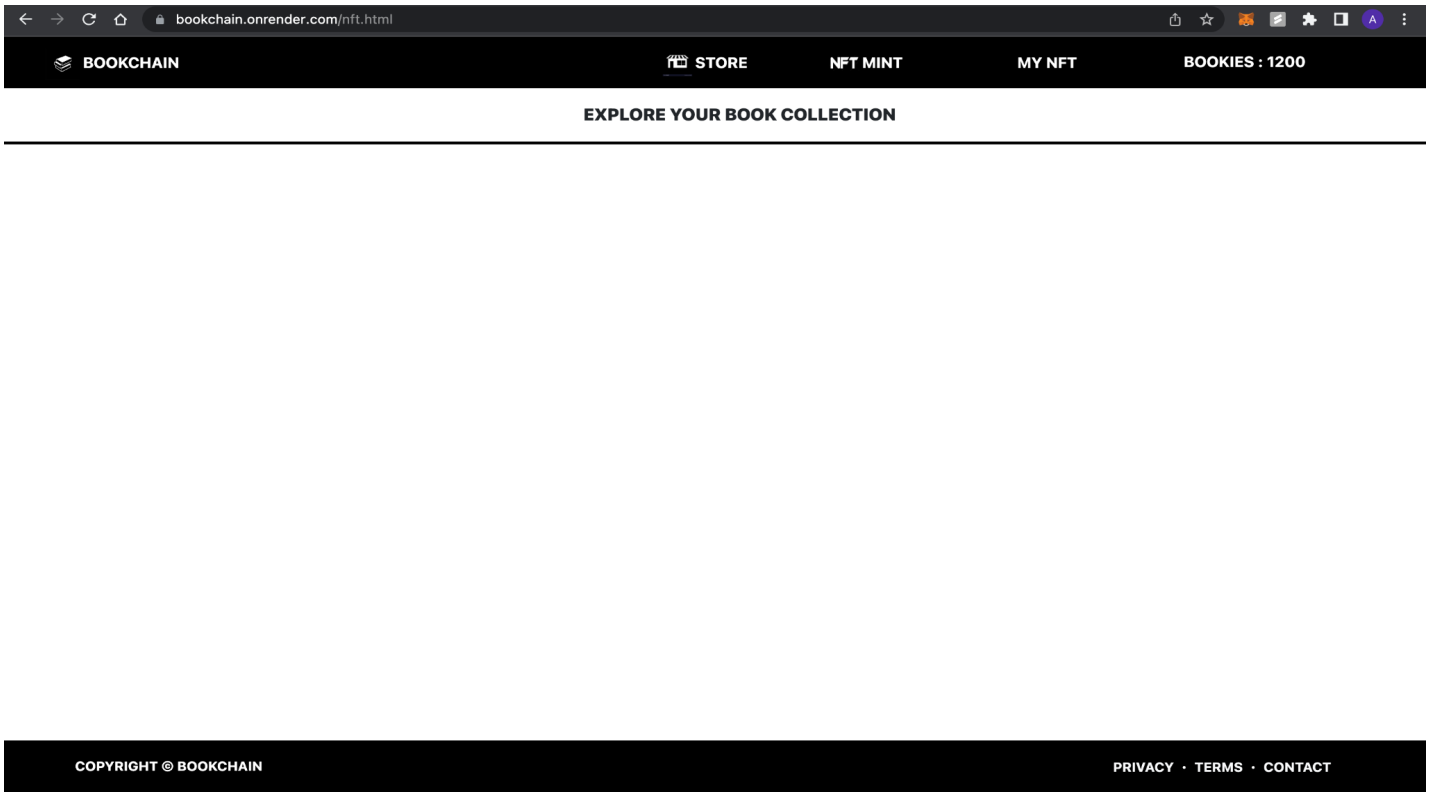
Once confirmed, the user is notified of the successful purchase of NFT via an alert from the application.



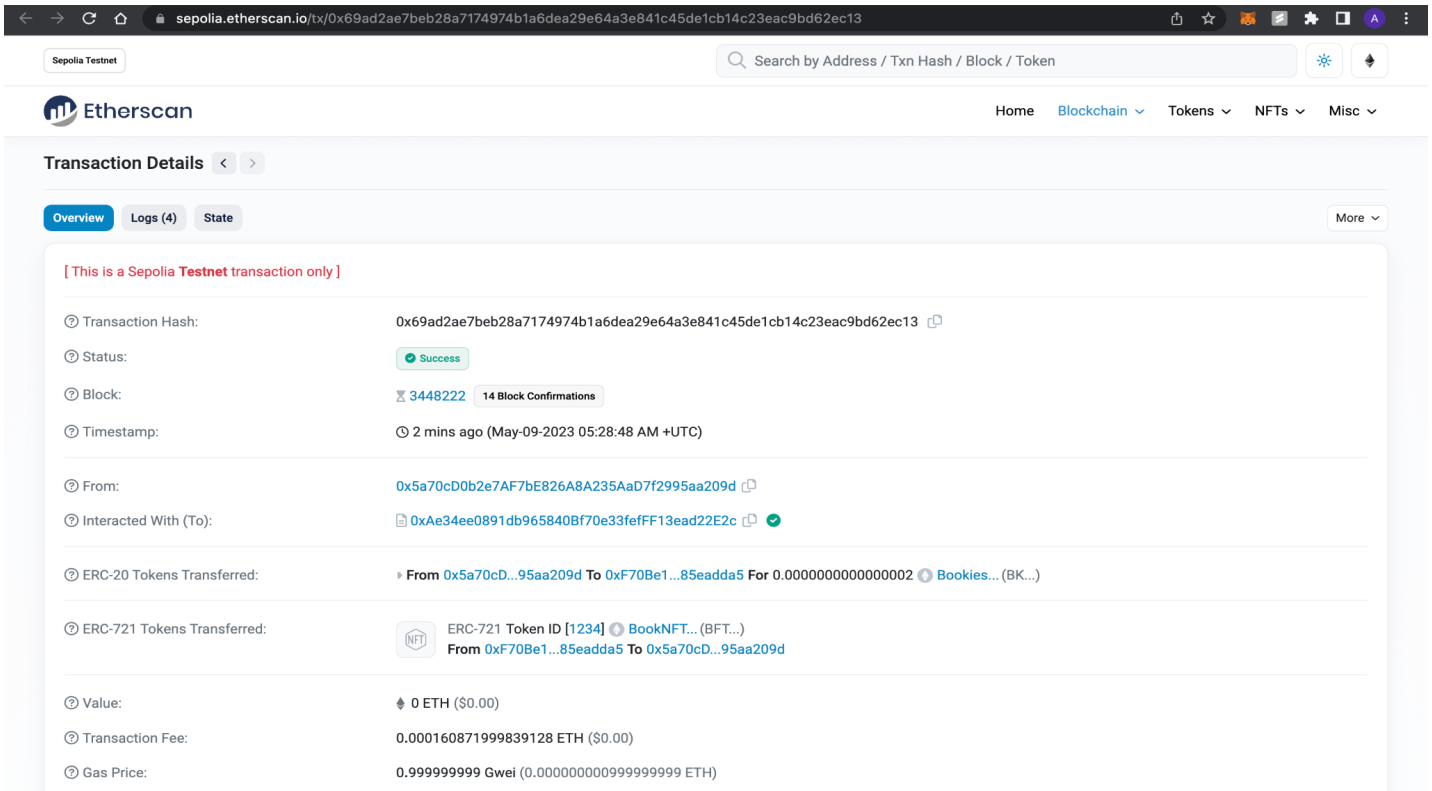
Now the purchased NFT is moved to his inventory. We can also note that the 200 Bookies used to purchase the NFT has been reduced from the buyer (user 2) account (Bookies : 800).



Similarly, the NFT is no longer in the inventory of the owner (user 1) account and the 200 Bookies for the purchase of NFT has been transferred to the owner (user 1) account (Bookies : 1200).



The transfer of NFT can also be verified from the Etherscan logs.



References

OpenZeppelin ERC20 :	https://docs.openzeppelin.com/contracts/2.x/erc20
OpenZeppelin ERC721 :	https://docs.openzeppelin.com/contracts/2.x/erc721
Ethereum ERC 20 :	https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
Ethereum ERC 721 :	https://ethereum.org/en/developers/docs/standards/tokens/erc-721/
Bootstrap :	https://www.w3schools.com/bootstrap5/
Mongoose :	https://mongoosejs.com/docs/
MongoDB :	https://api.mongodb.com/
Infura :	https://www.infura.io/
Render :	https://render.com/

Approved by Zhaofeng Si

Application URL : https://bookchain.onrender.com/