Karlsruher Institut für Technologie

NFT Marketplace

DOCUMENTATION

Prepared By:

David Blumenthal,
Tim Schaible,
Lukas Stingl,
Pablo Müller,
Alwin Fassbänder

DATE

13.04.2022

Table of Contents

INTRODUCTION

Non-Fungible Tokens (NFTs) are all over the media lately. As they are linked to one of the latest technology hypes namely the Metaverse and used in the massive computer games industry, NFTs can be seen as a huge potential for future business models. NFTs are an innovative type of digital asset. The digital token acquires value by proving the rightful ownership of a digital asset, such as a gaming item or digital art (image, audio, video). Within the scope of this "Blockchain Hackathon," we want to benefit from this new hype in distributed ledger technologies (DLT) and set up a platform where users all over the world can get engaged in NFT trades. Our main goal was to develop a decentralized NFT marketplace based on smart contracts powered by Ethereum to enable users to take advantage of the current NFT hype and participate in this new world of Non-Fungible Tokens. The platform will be an open and decentralized marketplace, where every user can mint his own NFTs, trade the newest NFTs and display their acquired collection for other users. It is also important for us that the platform supports one of the most common standards for NFTs (ERC-721, or ERC-1155).

Purpose

With the help of this document, we want to make it easier for the reader to not only understand the architecture of our software artefact but also get introduced to the features of the NFT marketplace. Further, the document provides the reader with all the necessary steps to install the software, set up the marketplace and participate in trades on the platform. We also want to indicate the steps of our work in a detailed timeline, introduce the reader to all the technologies used to support our development process and explain the approaches to our solution. In the end, we also point out further possible features that could be added in the future but simply go beyond the scope of this hackathon.

Timeline

After our first meeting and the clarification of basic project management questions like which communication channels to use, we started writing user stories and concluded them into the main requirements of our NFT marketplace - sorted by different priorities. Regarding project management, we decided to work with the "Issues" - feature provided by GitLab. There, we created Issues, provided them with a description and priority, and assigned them to a group member. We kept track of our issues by using the "Boards" - feature, where we divided the different issues into three categories – open, doing and closed. The next step for the team was to start with a simple click-dummy as a basis to add further features to this status quo in the future. In the 2nd week after our first meeting, we shared the work between two groups. Group one was responsible for the front-end development which means this group built the user interface of our website and designed the different pages of the NFT marketplace. Group two was responsible for the backend, particularly the writing of the smart contracts and the storage of the metadata through IPFS.
For a visualization of the timeline see Figure 1.

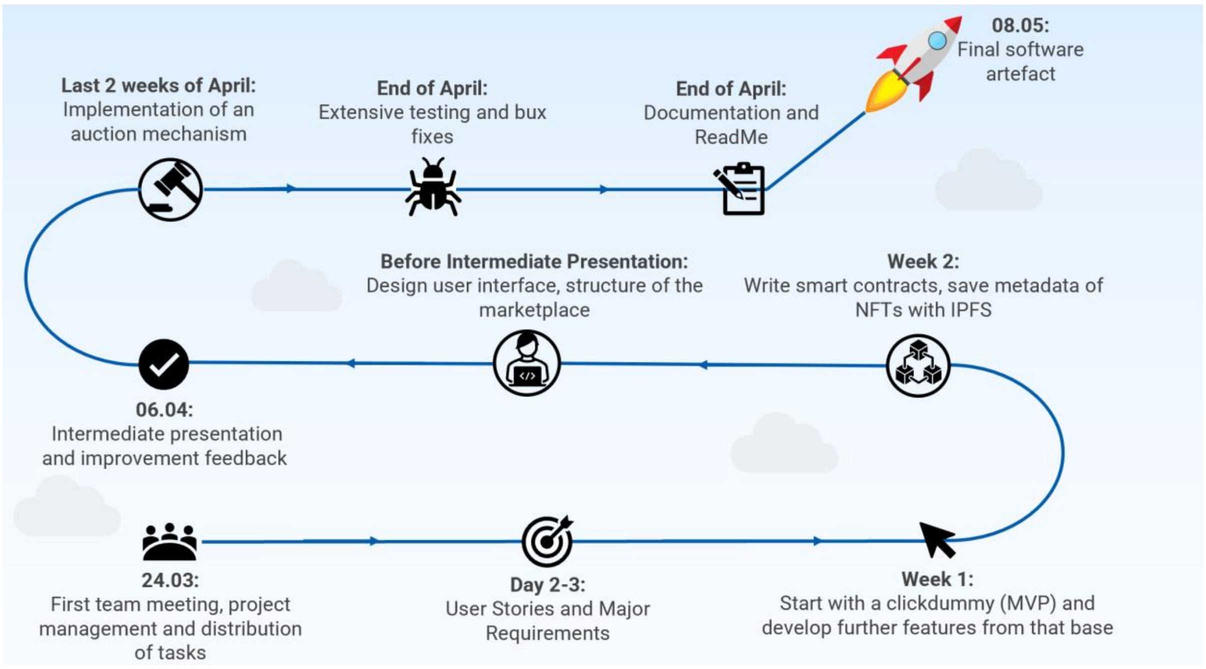

24.03:
First team meeting, project management and distribution of tasks
Day 2-3:
User Stories and Major Requirements
Week 1:
Start with a clickdummy (MVP) and develop further features from that base
Week 2:
Write smart contracts, save metadata of NFTs with IPFS
Before Intermediate Presentation:
Design user interface, structure of the marketplace
06.04:
Intermediate presentation and improvement feedback
Last 2 weeks of April:
Implementation of an auction mechanism
End of April:
Extensive testing and bux fixes
End of April:
Documentation and ReadMe
08.05:
Final software artefact

Figure 1 Outline of the project timeline

After we were able to show a fully working NFT marketplace at the intermediate presentation, we got valuable feedback on how to improve the functionality and user interface of our app. First, we started to implement an auction mechanism in addition to the "buy now button". Secondly, we wanted to enable the user to pay in different Ethereum-based currencies (abended due to lack of time see "room for possible improvement" on page 18). Further, we implemented indicators, assuring the user that a loading process is currently in progress. In the last weeks before the final submission of our software artefact, we focused on extensive testing and bug fixing, as well as on the creation of this documentation.

GETTING STARTED

Before we dive deeper into the set-up of the marketplace, as well as the more detailed functionality we introduce the used technologies in Table 1. We used these technologies to support our development process.

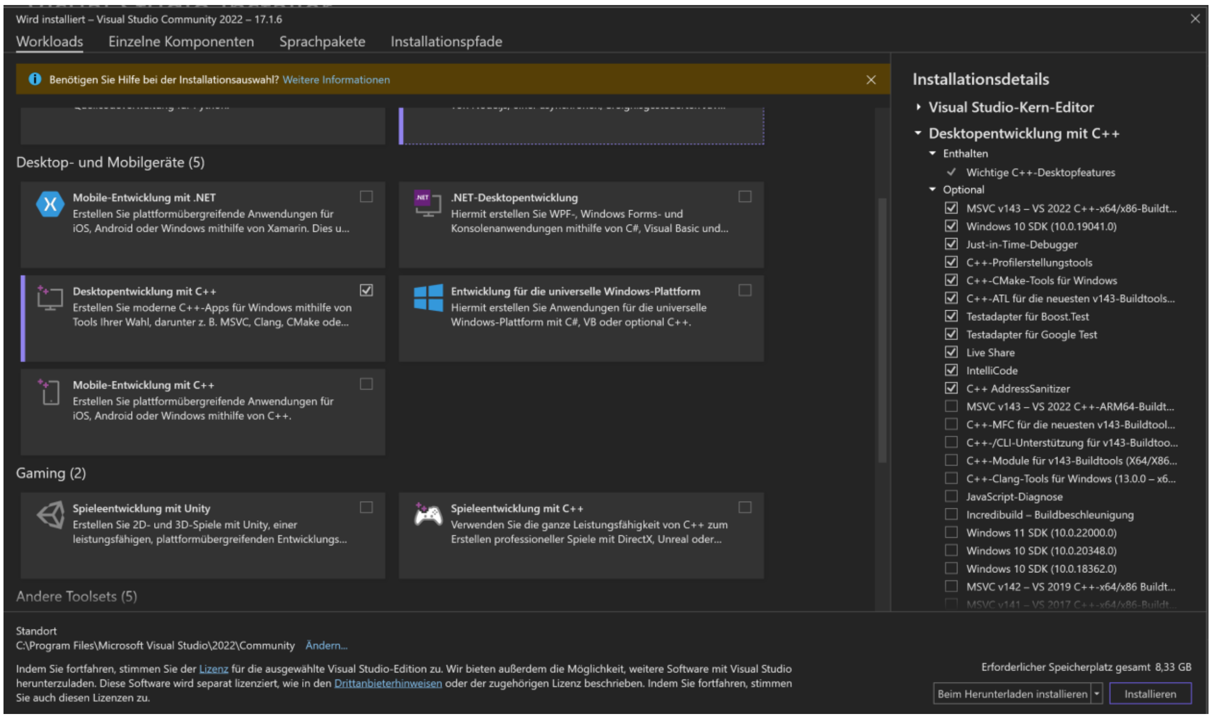
Table 1 Technologies used in the presented software artefact

	Solidity is an object-oriented, application-specific higher-level **programming language** for developing smart contracts for blockchain platforms like Ethereum or Tron, influenced by C++, Python, and JavaScript.
React JS	React is a JavaScript **software library** that provides a basic framework for outputting user interface components of web pages. We used it to develop the user interface of our marketplace.
	Web3.js is a collection of **JS libraries that allows interacting with an Ethereum node** using remotely or locally. Simply, it provides us with an API to use so we can easily work with the blockchain.
	"Kovan Test Network" is a **publicly accessible blockchain** for Ethereum. Using this public blockchain we can simulate a situation closer to the reality than by using five different, locally running Ganache blockchains.
node JS	Node.js is an open-source, cross-platform, backend **JavaScript runtime environment** built on Chrome's V8 JavaScript engine.
INFURA	Infura is a **blockchain development suite** which provides the tools for developers to easily access the Ethereum and IPFS network.
IPFS	IPFS (InterPlanetary File System) is a peer-to-peer hypermedia protocol that gets used to **store all the Metadata of the NFTs**. IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node that has it using a distributed hash table (DHT).
	Metamask is a **crypto wallet** that supports all types of Ethereum-based tokens. It can serve as a regular crypto wallet, but its strength lies in its connectivity to smart contracts and decentralized apps. It is also easy to use the Metamask Wallet on the "Kovan test chain" by simply selecting this pre-defined option in your wallet as the network.
TRUFFLE	Truffle is a **development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM)**, aiming to make life as a developer easier.

Prerequisites

To run the application, you'll need to install the following software (*click links*):

- Git
- Python 3
- Visual Studio 2022 with C++ Development extension (important to select C++ Desktop development while installing, see below)
- Node
- Truffle (see below: Installation of the Software)



Wird installiert – Visual Studio Community 2022 – 17.1.6

Workloads | Einzelne Komponenten | Sprachpakete | Installationspfade

Benötigen Sie Hilfe bei der Installationsauswahl? Weitere Informationen

Desktop- und Mobilgeräte (5)

- [ ] **Mobile-Entwicklung mit .NET** – Erstellen Sie plattformübergreifende Anwendungen für iOS, Android oder Windows mithilfe von Xamarin. Dies u...
- [ ] **.NET-Desktopentwicklung** – Hiermit erstellen Sie WPF-, Windows Forms- und Konsolenanwendungen mithilfe von C#, Visual Basic und...
- [✓] **Desktopentwicklung mit C++** – Erstellen Sie moderne C++-Apps für Windows mithilfe von Tools Ihrer Wahl, darunter z. B. MSVC, Clang, CMake ode...
- [ ] **Entwicklung für die universelle Windows-Plattform** – Hiermit erstellen Sie Anwendungen für die universelle Windows-Plattform mit C#, VB oder optional C++.
- [ ] **Mobile-Entwicklung mit C++** – Erstellen Sie plattformübergreifende Anwendungen für iOS, Android oder Windows mithilfe von C++.

Gaming (2)

- [ ] **Spieleentwicklung mit Unity** – Erstellen Sie 2D- und 3D-Spiele mit Unity, einer leistungsfähigen, plattformübergreifenden Entwicklungs...
- [ ] **Spieleentwicklung mit C++** – Verwenden Sie die ganze Leistungsfähigkeit von C++ zum Erstellen professioneller Spiele mit DirectX, Unreal oder...

Andere Toolsets (5)

Installationsdetails

- Visual Studio-Kern-Editor
- Desktopentwicklung mit C++
  - Enthalten
    - ✓ Wichtige C++-Desktopfeatures
  - Optional
    - [✓] MSVC v143 – VS 2022 C++-x64/x86-Buildt...
    - [✓] Windows 10 SDK (10.0.19041.0)
    - [✓] Just-in-Time-Debugger
    - [✓] C++-Profilerstellungstools
    - [✓] C++-CMake-Tools für Windows
    - [✓] C++-ATL für die neuesten v143-Buildtools...
    - [✓] Testadapter für Boost.Test
    - [✓] Testadapter für Google Test
    - [✓] Live Share
    - [✓] IntelliCode
    - [✓] C++ AddressSanitizer
    - [ ] MSVC v143 – VS 2022 C++-ARM64-Buildt...
    - [ ] C++-MFC für die neuesten v143-Buildtool...
    - [ ] C++-/CLI-Unterstützung für v143-Buildtoo...
    - [ ] C++-Module für v143-Buildtools (X64/X86...
    - [ ] C++-Clang-Tools für Windows (13.0.0 – x6...
    - [ ] JavaScript-Diagnose
    - [ ] Incredibuild – Buildbeschleunigung
    - [ ] Windows 11 SDK (10.0.22000.0)
    - [ ] Windows 10 SDK (10.0.20348.0)
    - [ ] Windows 10 SDK (10.0.18362.0)
    - [ ] MSVC v142 – VS 2019 C++-x64/x86 Buildt...
    - [ ] MSVC v141 – VS 2017 C++-x64/x86-Buildt...

Standort
C:\Program Files\Microsoft Visual Studio\2022\Community Ändern...

Indem Sie fortfahren, stimmen Sie der Lizenz für die ausgewählte Visual Studio-Edition zu. Wir bieten außerdem die Möglichkeit, weitere Software mit Visual Studio herunterzuladen. Diese Software wird separat lizenziert, wie in den Drittanbieterhinweisen oder der zugehörigen Lizenz beschrieben. Indem Sie fortfahren, stimmen Sie auch diesen Lizenzen zu.

Erforderlicher Speicherplatz gesamt 8,33 GB

Then you need to clone the repository by using the following command:

$ git clone https://git.scc.kit.edu/uflgi/nft-marketplace.git

Installation of the Software (also see ReadMe)

Now go to the project folder and run:

```
$ cd nft-marketplace

# install the dependencies
$ npm install

# install truffle
$ npm install truffle -g

# deploy de contracts on the kovan testchain
$ truffle migrate --reset --network kovan

# start the application
$ npm start
```

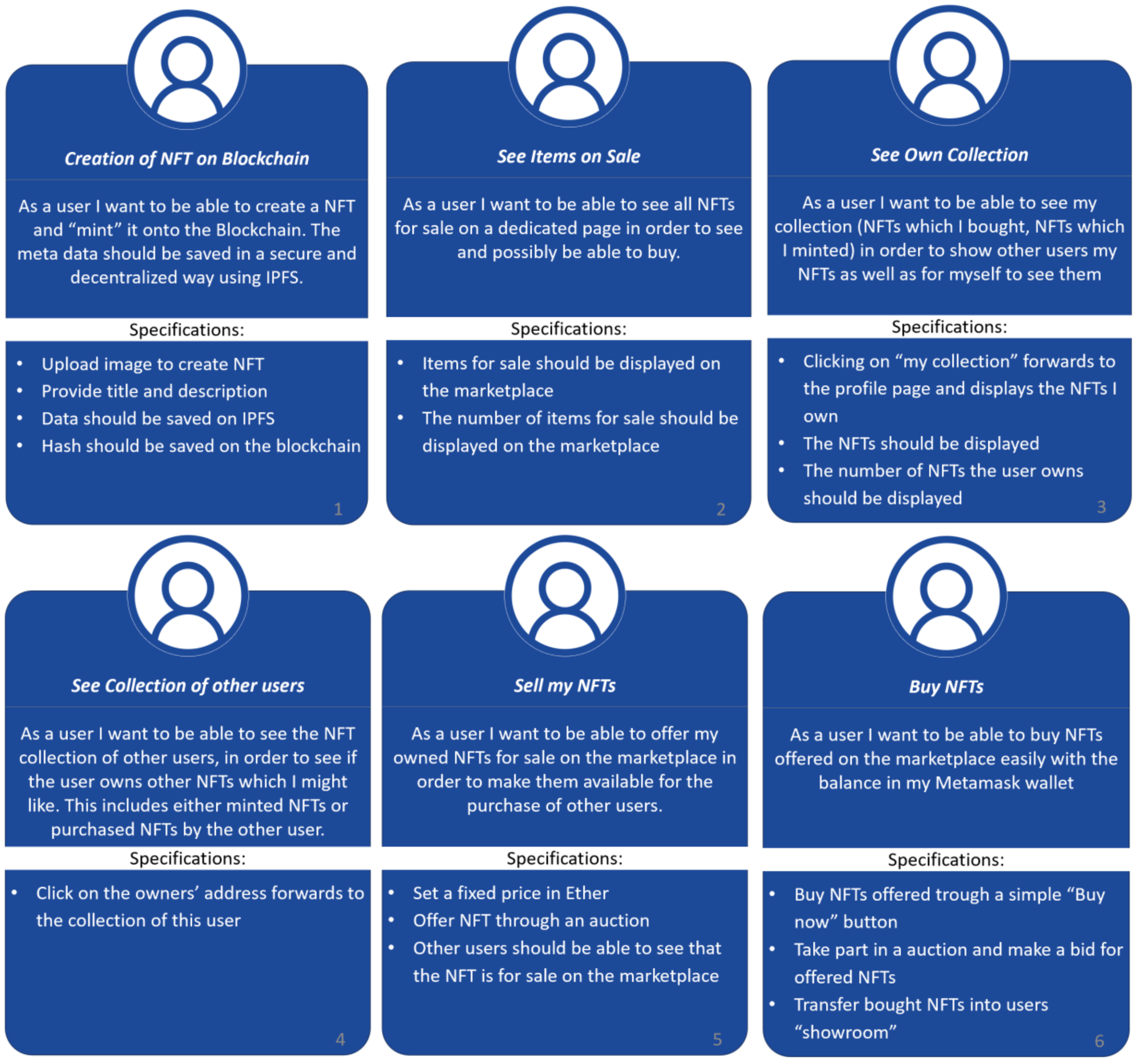
Before you can check out all the features of our marketplace, you need to follow a few steps to set up a wallet to pay for the transactions on the marketplace. Firstly, it is required to install Metamask wallet as a browser (we recommend using Google Chrome) extension: https://metamask.io/

Then you should configure Metamask to connect to the Kovan testchain. This requires the following:

- Open your Metamask wallet (click on Metamask in your browser extensions)
- Go to settings → advanced setting
- Scroll down and toggle the Button "show test networks" to on
- Open the Network Configuration panel
- Select the "Kovan Test Network"
- Get Free Kovan Ether on https://faucets.chain.link/

REQUIREMENTS OF A NFT MARKETPLACE

User stories - Functional Requirements



Creation of NFT on Blockchain

As a user I want to be able to create a NFT and “mint” it onto the Blockchain. The meta data should be saved in a secure and decentralized way using IPFS.

Specifications:

- Upload image to create NFT
- Provide title and description
- Data should be saved on IPFS
- Hash should be saved on the blockchain

1

See Items on Sale

As a user I want to be able to see all NFTs for sale on a dedicated page in order to see and possibly be able to buy.

Specifications:

- Items for sale should be displayed on the marketplace
- The number of items for sale should be displayed on the marketplace

2

See Own Collection

As a user I want to be able to see my collection (NFTs which I bought, NFTs which I minted) in order to show other users my NFTs as well as for myself to see them

Specifications:

- Clicking on “my collection” forwards to the profile page and displays the NFTs I own
- The NFTs should be displayed
- The number of NFTs the user owns should be displayed

3

See Collection of other users

As a user I want to be able to see the NFT collection of other users, in order to see if the user owns other NFTs which I might like. This includes either minted NFTs or purchased NFTs by the other user.

Specifications:

- Click on the owners’ address forwards to the collection of this user

4

Sell my NFTs

As a user I want to be able to offer my owned NFTs for sale on the marketplace in order to make them available for the purchase of other users.

Specifications:

- Set a fixed price in Ether
- Offer NFT through an auction
- Other users should be able to see that the NFT is for sale on the marketplace

5

Buy NFTs

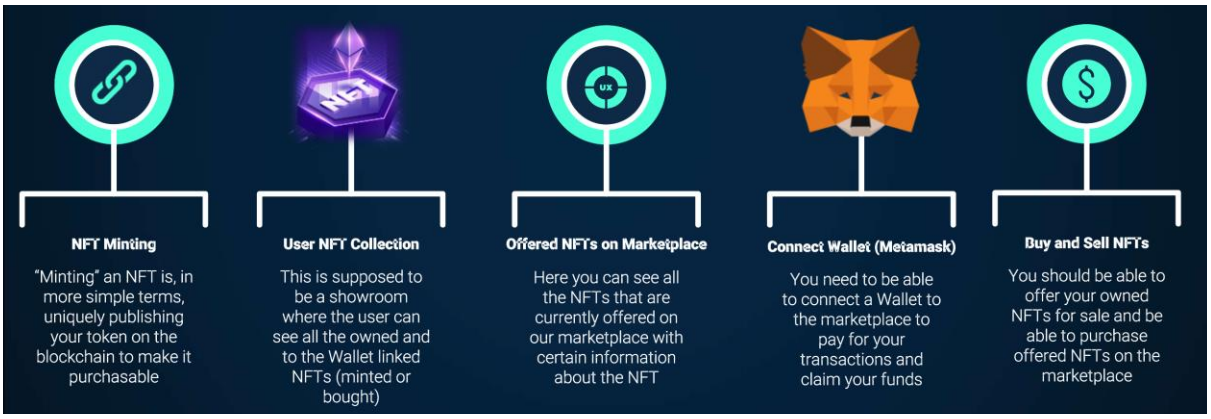
As a user I want to be able to buy NFTs offered on the marketplace easily with the balance in my Metamask wallet

Specifications:

- Buy NFTs offered trough a simple “Buy now” button
- Take part in a auction and make a bid for offered NFTs
- Transfer bought NFTs into users “showroom”

6

Major Requirements



NFT Minting

"Minting" an NFT is, in more simple terms, uniquely publishing your token on the blockchain to make it purchasable

User NFT Collection

This is supposed to be a showroom where the user can see all the owned and to the Wallet linked NFTs (minted or bought)

Offered NFTs on Marketplace

Here you can see all the NFTs that are currently offered on our marketplace with certain information about the NFT

Connect Wallet (Metamask)

You need to be able to connect a Wallet to the marketplace to pay for your transactions and claim your funds

Buy and Sell NFTs

You should be able to offer your owned NFTs for sale and be able to purchase offered NFTs on the marketplace

APPLICATION, ARCHITECTURE AND STRUCTURE

The NFT Marketplace follows a three-layer architecture. The bottom layer is the blockchain part itself consisting of the actual chain, the Ethereum Virtual Machine, and the Smart Contracts, that get interpreted and executed by it. Further, the metadata (the file itself, name, type, and description) of each NFT is saved leveraging IPFS, a peer-to-peer hypermedia protocol. The user can access the application via a web browser (with Metamask wallet installed). The user interface, built with React.js, relies on the web3.js library to communicate with the smart contracts through Metamask. Meaning that the data reflected on the front-end application is fetched from the Ethereum blockchain. Each action performed by the user (mint an NFT, offer NFT, buy NFT...) creates a transaction on the Ethereum blockchain and therefore requires a Metamask confirmation and the payment of a small fee (gas costs). Each transaction will permanently modify the state of the NFTCollection and NFTMarketplace smart contracts. On top of it, the user will upload the NFT Metadata to the IPFS, creating a permanent hash which will be recorded on the blockchain itself to prove ownership. for a graphical representation of the above-described architecture and processes see Figure 2, Figure 3 and Figure 4.

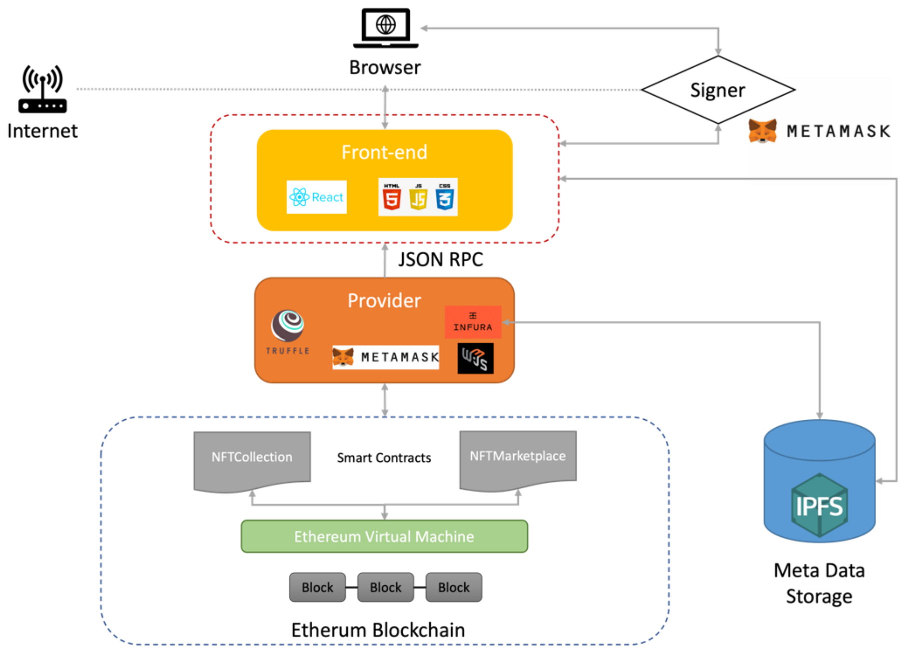


Figure 2 Architectural diagram of the NFT marketplace

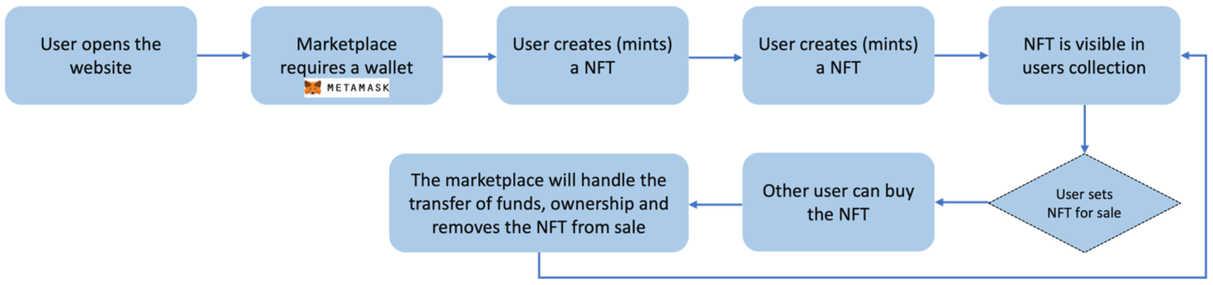

User opens the website
Marketplace requires a wallet
METAMASK
User creates (mints) a NFT
User creates (mints) a NFT
NFT is visible in users collection
User sets NFT for sale
Other user can buy the NFT
The marketplace will handle the transfer of funds, ownership and removes the NFT from sale

Figure 3 Flowchart of user interaction

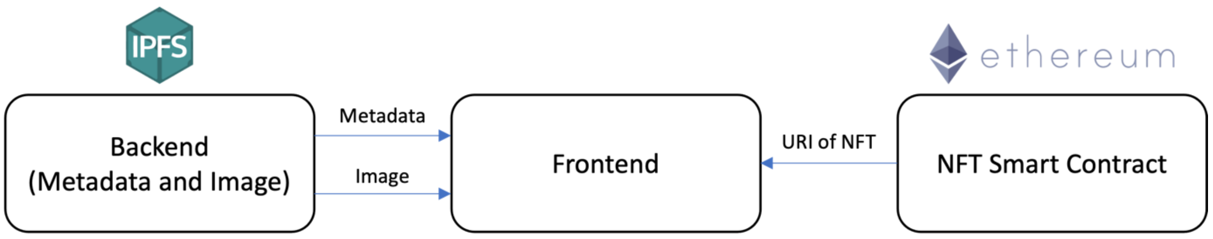


Figure 4 Outline of backend architecture

Components

The Components folder (in the source code) holds the functionality of the application. Each of the described modules below can be found with the same name in the provided source code.

Content

Collection

The Collection module retrieves NFTs from the blockchain according to whether the collection on which the user "clicked" is the user's collection or another user's collection. Further, it displays the retrieved NFTs with their associated metadata.

Create

The Create module is responsible to provide the functionality for a user to be able to create their NFT(s). The user can upload an image and provide a title and description. If the user decides to create an NFT a hash for it is created and written onto the blockchain. Further, the provided metadata will be saved leveraging the IPFS protocol.

Marketplace

The marketplace retrieves NFTs from the blockchain and checks if an NFT is for sale or not. Only NFTs which are on sale are then displayed with their title, owners' address, image, and their respective price. The user has the option to buy an NFT or to further inspect it in greater detail. The Detail Page is an isolated view of the NFT under inspection and displays apart from before mentioned metadata the description.

Not Found

The Not Found displays a page saying 404 Error, which occurs if the user for arbitrary reasons sends a request (URL) which is not backed by the application.

Smart Contracts

NFT Collection

The NFT Collection smart contract is responsible for mapping and creating an NFT on the blockchain. It inherits and enforces the ERC-721 Non-Fungible Token Standard from Ethereum. With the ERC-721 Standard, we can rely on many useful, already implemented functionalities The module provides the following functionalities and events as displayed in Table 2 and Table 3.

Table 2 Functionalities of the NFT collection smart contract

safeMint	Used to save the tokenURI of the NFT in the blockchain. The tokenURI consists of the unique hash value, returned from IPFS when uploading a file
getPastEvents	Returns all events for a specified transaction, for example the creation of an NFT
tokenURI	Returns the IPFS hash string from the blockchain, when providing the tokenId as an integer
ownerOf	Returns the address of the owner, when providing tokenId of NFT
approve	transfers the NFT from the owner to an address, when providing the tokenId and the receiver address

Table 3 Events of NFT collection smart contract

Transfer	Is emitted, once a Transfer is made on the blockchain

NFT Marketplace

The NFT Marketplace smart contract comprises the main functionality. This includes setting the status of an NFT to sale and reversing this status back to “not on sale”. Additionally, the smart contract handles the sales of NFTs from one user to the other, including the proper transfer of tokens (price). In addition, the NFT Marketplace smart contract delivers functionality to offer smart contracts in an English Auction. The auction time is fixed (5 minutes for testing purposes -> function can be extended to receive auction time as well from frontend -> chapter 7). For further detail of the smart contract’s functionalities and events see Table 4 and 5.

Table 4 Functionalities of the NFT marketplace smart contract

offerCount	Returns number of total created offers
auctionCount	Returns number of total created auctions
offers	Returns offer struct, when receiving offerId
auctions	Returns auction struct, when receiving auctionId
userFunds	Returns how much funds each account has stored in the smart contract, when providing an address
claimFunds	Transfers funds from smart contract to user wallet
makeOffer	Transfers the NFT from seller to smart contract and emits Offer Event, when providing the NFT ID and price
makeAuction	Transfers the NFT from seller to smart contract and emits Auction Event, when providing the NFT ID and price
fillOffer	Transfers the NFT from the smart contract to the buyer, when providing an

	offerId; adds price to userFunds; checks that offer is existing and valid, buyer is not the owner and price equals price set in makeOffer; emits OfferFilled event
fillBid	Sets bids of user to msg.value; returns previous highest bid to previous highest bidder; checks that there is a valid auction; checks that the seller cannot bid on his own NFT; checks that bid is higher than previous highest bid (see autoBookBack)
setBids	Sets bid of bidder in auction struct, when providing bidder and bid
cancelOffer	The offer is canceled, when offerId is provided; checks that user is owner of the NFT and offer is still valid; emits OfferCancelled Event
autoBookBack	When a user fills a new bid and becomes the highest bidder, the bid of the previous highest bidder is transferred back to the wallet of the previous highest bidder
end	end is called after the auction has ended and the “Redeem NFT” – button was clicked by the highest bidder; if someone bid on the NFT, it is transferred to the highest bidder, else transferred back to the seller; emits Auction successful; checks if auction is valid

Table 5 Events of NFT marketplace smart contract

OfferFilled	Is emitted once an offer is made. Returns offerId, NFT ID and newOwner
OfferCancelled	Is emitted once an offer is canceled. Returns offerId, NFT ID and owner
ClaimFunds	Is emitted once a user claims his available funds; returns address of user and amount
AuctionCreated	Is emitted once an auction is made; returns auctionId, NFT ID, startingPrice and duration
AuctionSuccessful	Is emitted once an auction is finished successfully; returns auctionId, NFT ID, highestBid and highestBidder

Once the contracts are compiled and migrated using truffle, they can be verified and published using a truffle extension “truffle-plugin-verify” that can be installed via npm. Using the command “truffle run verify NFTMarketplace –network kovan” verifies the NFTMarketplace smart contract, deployed on the Kovan testchain and publishes the contract to Etherscan.io. For **testing** and development purposes we used the tool **Remix IDE**, which allows developing, deploying and administering smart contracts for Ethereum like blockchains. With Remix IDE we additionally gained the possibility to debug our smart contracts, which was very helpful during our development process. Once we had a reasonable and functional smart contract, we started to adapt our frontend to the new functionalities implemented in our backend.

Regarding **security**, we encountered some possible areas of attacks. We are aware that there is a possibility in the auction mechanism where a user can prevent to get overbidden. By using a smart contract that bids on the item, but does not allow to take back ETH, a revert gets thrown in the *autoBookBack* when another user tries to place a higher bid on the item. This is because the smart contract implemented by the other user does not take back any ETH. This is something you should definitely keep in mind for further implementations.

Helpers

Holds outsourced helper functionalities such as formatting displayed prices to a certain number of decimals.

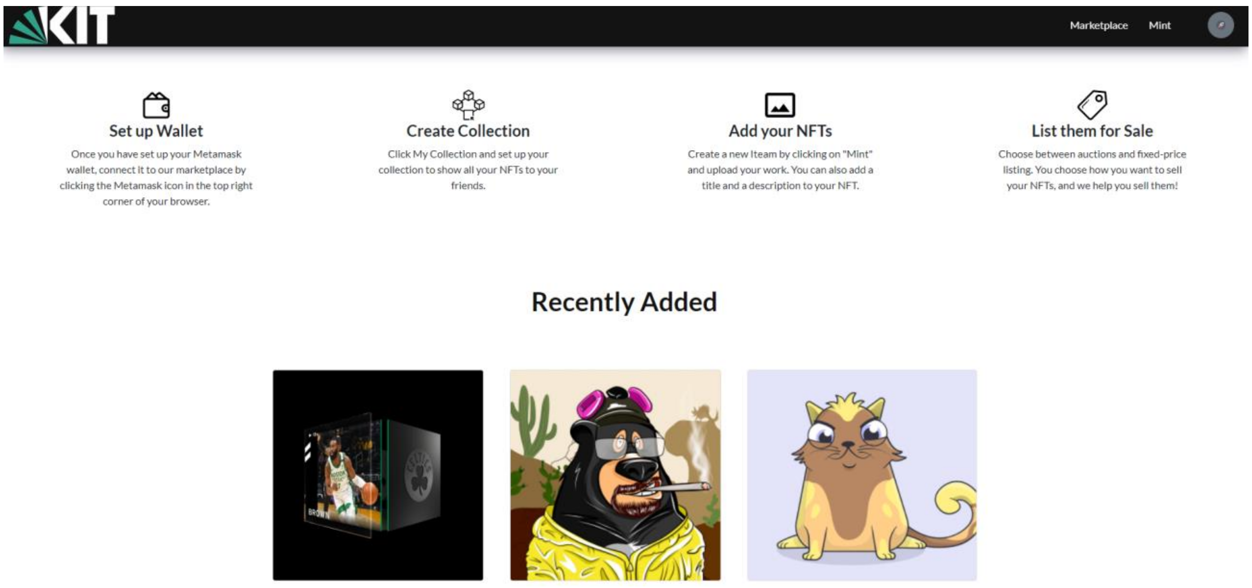
SYSTEM FEATURES

This chapter provides descriptions of the applications' features. In order to get a better understanding of the user interface screenshots are provided.

File Types:
When minting an NFT the user must not only input a name and a description but also upload a picture of the NFT as .png, .jpg, .jpeg.

Securely Stored Metadata:
The NFT Marketplace stores and pins the ERC-721 metadata as decentralized as feasible using Interplanetary File System (IPFS).

Latest drops:
On the "home" page you do not only get introduced to the main features of our marketplace but can also admire the three most recently added NFTs, which means this section shows the three latest NFT mints made by any user of the platform.



Set up Wallet

Once you have set up your Metamask wallet, connect it to our marketplace by clicking the Metamask icon in the top right corner of your browser.

Create Collection

Click My Collection and set up your collection to show all your NFTs to your friends.

Add your NFTs

Create a new Iteam by clicking on "Mint" and upload your work. You can also add a title and a description to your NFT.

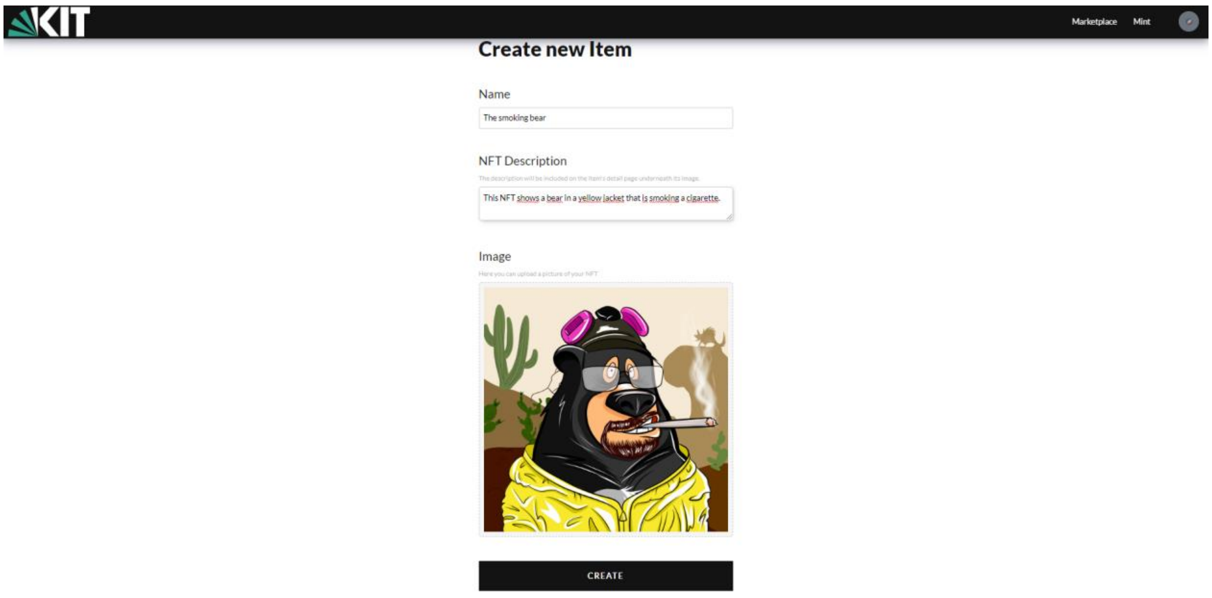
List them for Sale

Choose between auctions and fixed-price listing. You choose how you want to sell your NFTs, and we help you sell them!

Recently Added

Figure 5 Home page of the marketplace

<u>Mint NFTs:</u>



Create new Item

Name

The smoking bear

NFT Description

The description will be included on the item's detail page underneath its image.

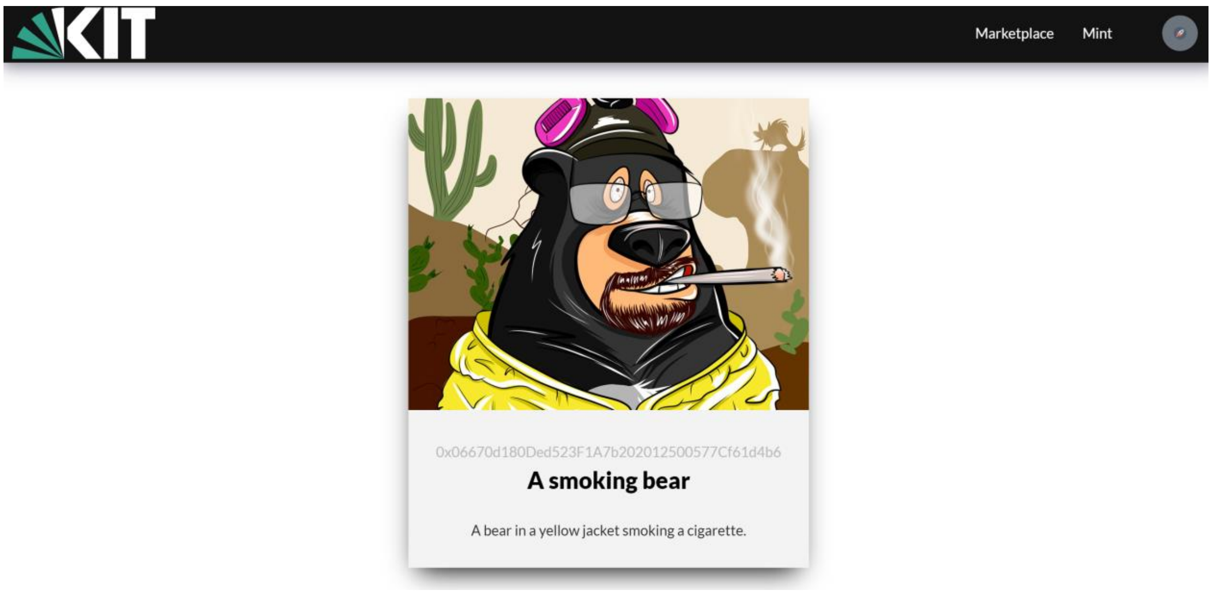
This NFT shows a bear in a yellow jacket that is smoking a cigarette.

Image

Here you can upload a picture of your NFT

Figure 6 Page for NFT creation

The user must input a name, description and upload a file (image) to mint his own NFT. Once minted, a representation of this NFT will be displayed in the collection of the creator and initially, it will be owned by its creator. This is open to everyone, meaning everyone can participate in this NFT creation within this collection. After the user confirms the minting transaction, the newly created "NFT card" gets displayed to the user (see Figure).



0x06670d180Ded523F1A7b202012500577Cf61d4b6

A smoking bear

A bear in a yellow jacket smoking a cigarette.

Figure 7 Detail page of novel NFT after creation

<u>Offer on the marketplace:</u>

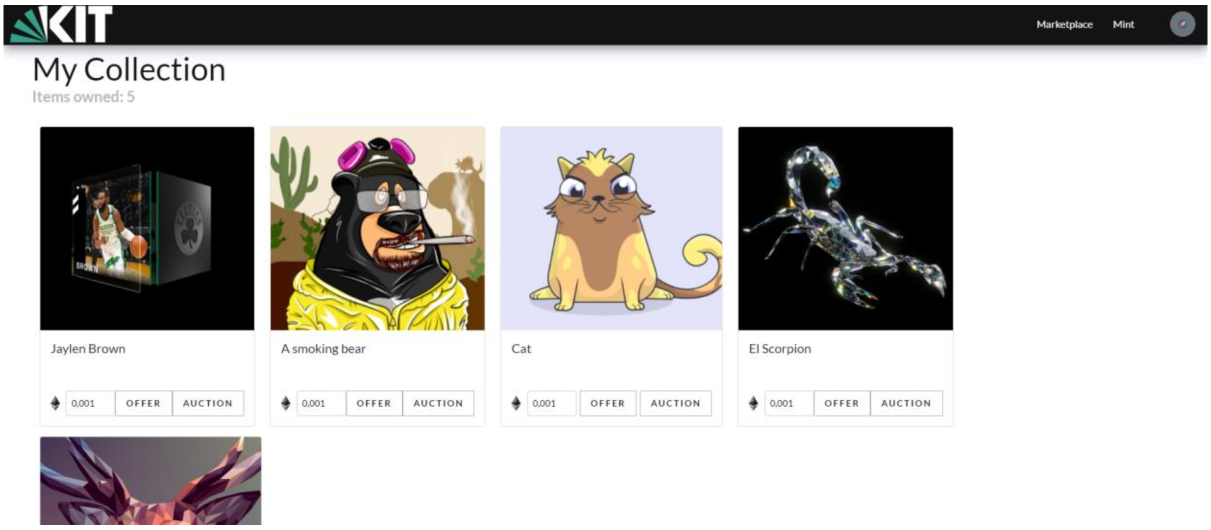


Figure 8 My collection page; offer a NFT via auction

The user can offer his NFTs (either bought or minted NFTs owned) by specifying their price (minimum 0.001 Ether). If someone fulfills this offer, then the ownership is transferred to a new owner in exchange for the price paid. If the owner of the offered NFT would like to cancel the sale, he can simply cancel the offer and add the NFT back into his collection.

The other option the user has is to offer the NFT through an auction. As soon as the NFT auction expires after a certain time (300 seconds in our case), the NFT gets transferred to the highest bidding user in exchange for the bid (in ETH). If there is not a single bid for the offered NFT the user can “redeem NFT” (see picture below) to add the unsold NFT back the personal collection.

<u>Buy NFT:</u>

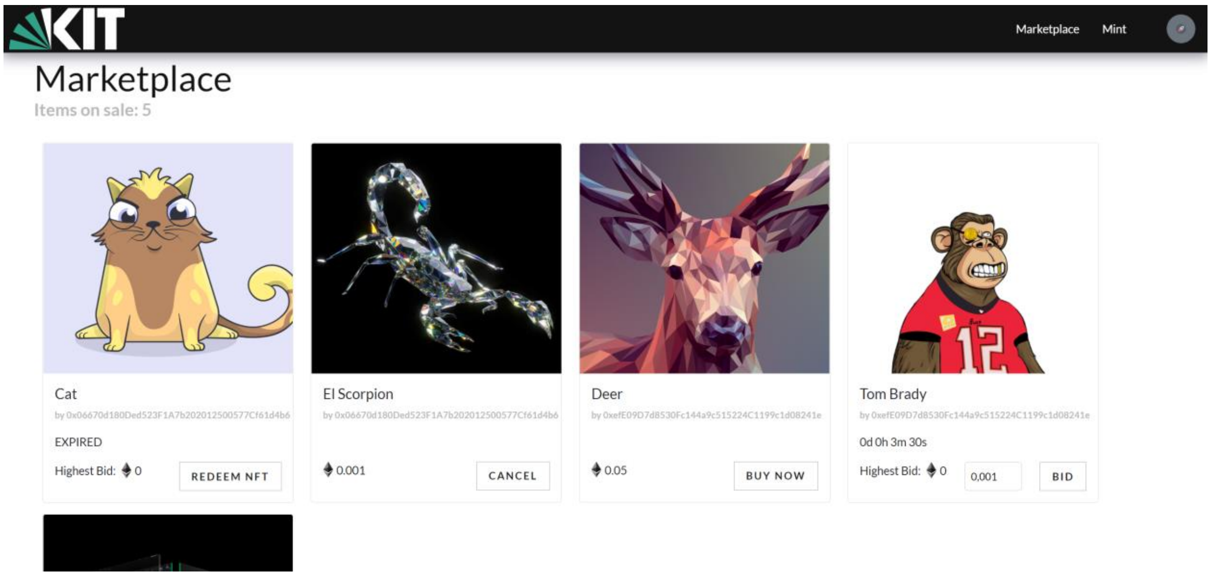
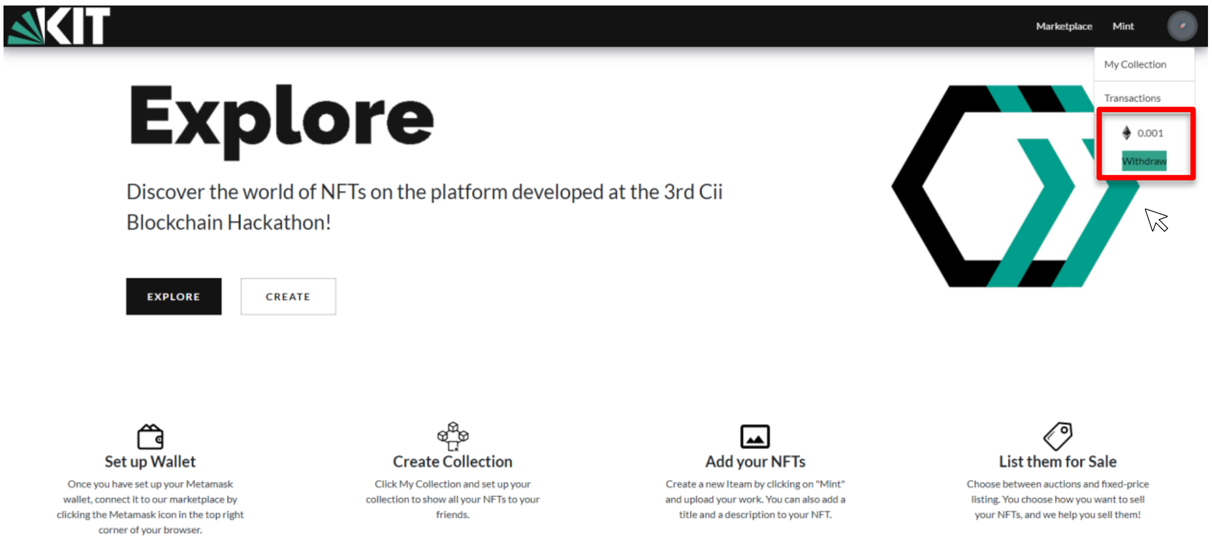

Marketplace
Mint
Marketplace
Items on sale: 5
Cat
by 0x06670d180Ded523F1A7b202012500577Cf61d4b6
EXPIRED
Highest Bid: 0
REDEEM NFT
El Scorpion
by 0x06670d180Ded523F1A7b202012500577Cf61d4b6
0.001
CANCEL
Deer
by 0xefE09D7d8530Fc144a9c515224C1199c1d08241e
0.05
BUY NOW
Tom Brady
by 0xefE09D7d8530Fc144a9c515224C1199c1d08241e
0d 0h 3m 30s
Highest Bid: 0
0,001
BID

Figure 9 Marketplace with offered NFTs

A user can buy the NFTs that someone else offered on the marketplace. This will require paying the requested price (the Ether will be transferred to the smart contract to be claimed later). Another option for the user is to make a bid for a NFT that is offered as an auction on the market. The user has also the option to cancel his/her offered NFT on the marketplace and replace it in the own collection.

Claim Funds:



Explore

Discover the world of NFTs on the platform developed at the 3rd Cii Blockchain Hackathon!

Set up Wallet

Once you have set up your Metamask wallet, connect it to our marketplace by clicking the Metamask icon in the top right corner of your browser.

Create Collection

Click My Collection and set up your collection to show all your NFTs to your friends.

Add your NFTs

Create a new Iteam by clicking on "Mint" and upload your work. You can also add a title and a description to your NFT.

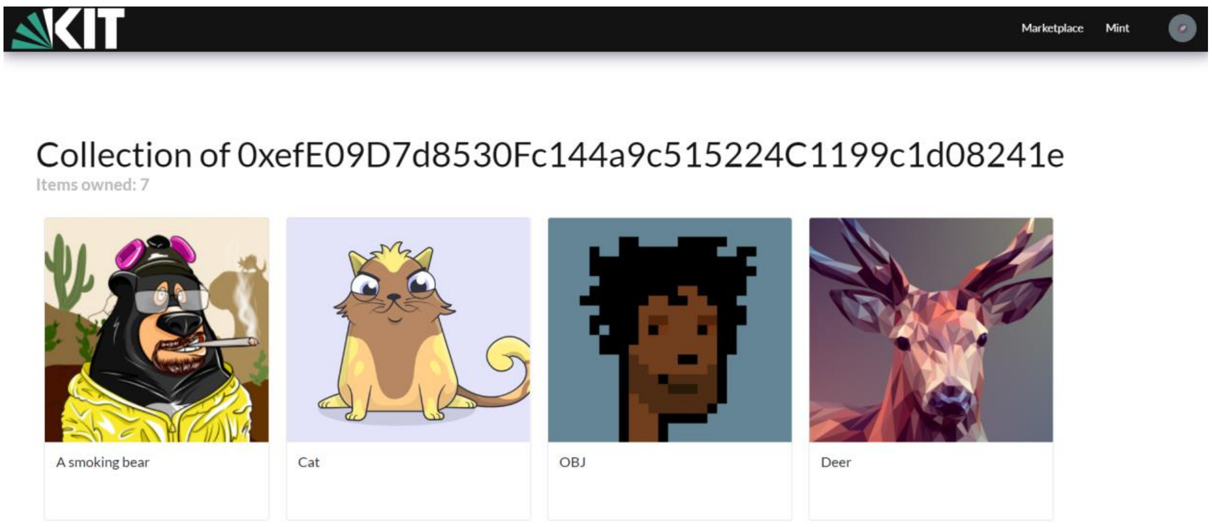
List them for Sale

Choose between auctions and fixed-price listing. You choose how you want to sell your NFTs, and we help you sell them!

Figure 10 Claim function to withdraw earned money

If a user has sold a NFT, he could claim his funds by clicking the "withdraw" button in the top-right corner. The funds then get transferred back to the related Metamask wallet.

See user Collection:



Collection of 0xefE09D7d8530Fc144a9c515224C1199c1d08241e

Items owned: 7

Figure 11 collection of another user

If you click on the private key of an NFT owner, you can access his/her showroom and have a look at the users NFT collection. You can also have a look at your own collection (see “make offer”).

Authentication

There is no external authentication necessary. When navigating to the user interface of the application, the web3 library checks if a Wallet (e.g., Metamask) is connected. If there is no wallet connected, the page does not load, and the user must connect a valid wallet to use the functionalities of the page.

ROOM FOR POSSIBLE IMPROVEMENTS

There is always room for improvements in software. In the following, we will reflect on non-functional and functional requirements that did not get implemented or not to the planned extent.

Non-functional requirements

The application has a relatively high standard in terms of **security** due to the inherent characteristics of the blockchain. As Metamask is used by the user as an authentication service, the user is responsible for securing his recovery phrase. Without the recovery phase, it is not possible to recover your account if you forgot your password.[1] If it is later intended to deploy our smart contracts onto the Ethereum main-net, more security checks should be implemented to secure the contracts against attacks (e.g., re-entrancy, front-running). **Performance** is currently the application's weakest point. The user will experience loading times that relative to classical websites might be considered as long. Those loading times occur when the NFTs are loaded. The unique ids of the NFTs are fetched relatively quickly. However, getting the corresponding metadata (mostly the image itself) causes those long loading times due to the decentralized peer-to-peer design of the IPFS protocol. **Maintainability** in terms of adding new functionality or changing the source code (smart contracts) in general, will be expensive once the application is deployed, due to high gas costs. Although it is not possible to offer a NFT a second time, while the first offer transaction is still waiting to be processed, the user gets not prevented by the UI to offer the same NFT for a second time. Regarding **Usability** in case of user error protection, this is something that needs further improvements. Additionally, it has to be mentioned that sometimes there occurs an irregularly TypeError while trying to cancel an existing offer. The functionality behind this canceling process fully works. When the user refreshes the website, the error disappears and one can continue to use the website as before.

[1] Basic Safety and Security Tips for MetaMask, Metamask Website, 25 March 2022

Functional requirements

Additional file types. Currently, our marketplace as described above only supports image filetypes. It is possible to have NFTs linked to other filetypes, such as sound (e.g. mp3), videos (mp4) or even 3D models (glTF). **Past NFT owners.** Currently, it is not possible to click on an NFT and display all the past transactions. Such a transaction history might contain information like from whom to whom was the NFT sold, when was it sold, and at which price was it sold. It is further possible to display the past price developments in charts inspired by the stock market. Further, we would have liked to implement more features and personalization options on the **profile page**. The showroom for instance is not yet customizable in the sense that the order of NFTs cannot be changed, and it is not possible to only display a curated selection of owned NFTs. Lastly, we wanted to implement a **transaction fee.** This should serve to operate the marketplace profitably or at least cover its costs. For each transaction sale/purchase, for example, 1.5% of the purchase price would have been diverted to an address associated with the Marketplace. Currently it is not possible for the user to select a **personalized time span** or an individual **starting price for his auction**. Every NFT that is offered through an auction is available for a specific time and for a specific starting price that we set fix in our code. One final improvement is **authentication.** The implementation of a central user register where a mapping between the signed-up user and his wallet address takes place could not only improve the usability but also the value of the application from a business perspective. For example, through the acquisition of e-mail addresses the field of marketing and newsletters might be an opportunity to pursue.

RESOURCES

- Ethereum: [https://ethereum.org/en/]
- IPFS: [https://ipfs.io/]
- ReactJS: [https://reactjs.org/]
- Kovan Testnet: [https://kovan-testnet.github.io/website/]
- Web3.js: [https://web3js.readthedocs.io/en/v1.7.3/]
- Node.js: [https://nodejs.org/en/]
- Metamask: [https://metamask.io/]
- Infura: [https://infura.io/]
- Solidity: [https://docs.soliditylang.org/en/v0.8.13/]
- Remix IDE: [https://remix.ethereum.org/]
- TruffleSuite: [https://trufflesuite.com/]
- Cryptozombies: [https://cryptozombies.io/en/course]