COLUMBIA UNIVERSITY

NFT Minter Full-stack DAPP

The project report of ELEN 6883: Introduction to Blockchain Technology

Xin Huang | UNI: xh2510
Ziqi Zhang | UNI: zz2881
Weicong Zhang | UNI: wz2582
Zhengyu He | UNI: zh2480
Kairui Yin | UNI: ky2483

Contents

Abstract

In this project, we program a full-stack blockchain DAPP mainly focusing on NFT minting, we use solidity to program the smart contract as the core of the backend and JS for the web development of the frontend, also we discuss a new encryption approach for Gifs and Video, which is inspired by the concept of Merkle Tree. By combining those techniques, we bring an DAPP with a core functionality of NFT minting and several novelties.

Keywords: Blockchain, Ethereum, Non-fungible token(NFT), Decentralized Application(DAPP), Merkle Tree

1 Introduction

Blockchain was first introduced in 1991 by Stuart Haber and W. Scott Stornetta [1]. First application of blockchain is a distributed ledger. Since Satoshi Nakamoto proposed the concept of Bitcoin [2] in 2008, Blockchain has been a hot topic. Envisioned by Bitcoin, Ethereum was conceived in 2013 by programmer Vitalik Buterin [3], then people and community invented many Decentralized applications (DAPP) like SocialFi, GameFi and NFT exchange. Based on the idea of Blockchain, we developed a DAPP which supports online non-fungible token (NFT) minting.

1.1 Bitcoin

Bitcoin is a completely digital, decentralized, engineered currency designed from using cryptography, computer science and economics. It offers the promise of an online currency that is secured without any central authority, unlike government-issued currencies. As a cryptocurrency, Bitcoin is quite successful and its technology is mature.

1.2 Ethereum and NFT

Although most applications of Blockchain is cryptocurrency, the Blockchain indeed has much more potential and it never stops evolving. Ethereum was born out as a new public blockchain in 2013 [3] with added functionalities compared to Bitcoin, a development that has turned out to be a pivotal moment in Blockchain history. Ethereum intends to provide a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create any applications, namely, smart contracts which is simply written the in a few lines of code.

1.2.1 Smart Contracts

Smart contracts are digital contracts stored on a blockchain that are automatically executed when predetermined terms and conditions are met.

1.2.2 Non-fungible Token

Non-fungible token (NFT) is a smart contract and it acts as a non-duplicable digital certificate of ownership for any assigned digital asset. NFTs can be everything. They can be a jpg image, music, or digital art and most NFTs are part of the Ethereum Blockchain.

1.3 Overview of Our DAPP

Inspired by the development of DeFi and NFT, in this project, we are trying to build up a full-stack blockchain DAPP about NFT minting, we will use solidity to program the smart contract to support the functionalities of NFT minting and JS for the front-end web development, which will be introduced in Section 3. Also, we aim to bring dynamic not only static image to NFT, which is to say we are bringing Gifs and videos to NFT, which is an originality of this project, and this will be introduced in Section 4.

In addition, the showcase of our DAPP can be found in Section 2, including screenshots, github repo and video demo.

2 Showcase

2.1 Screenshots

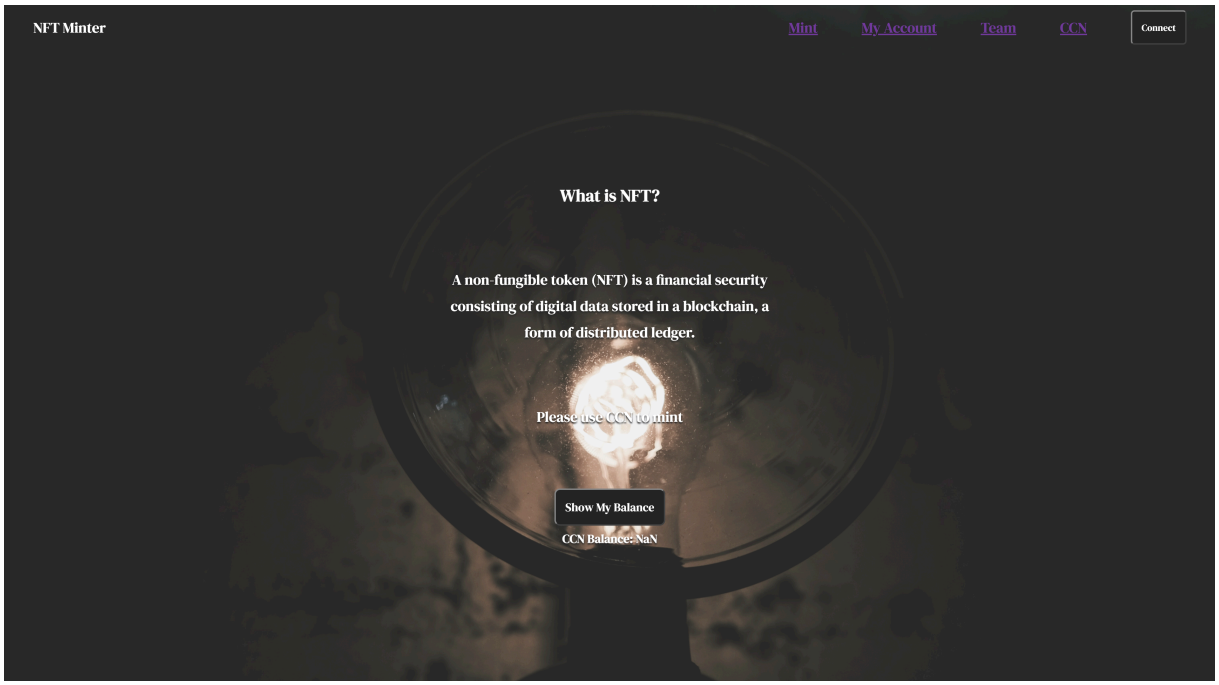

Figure 1: Main Page.

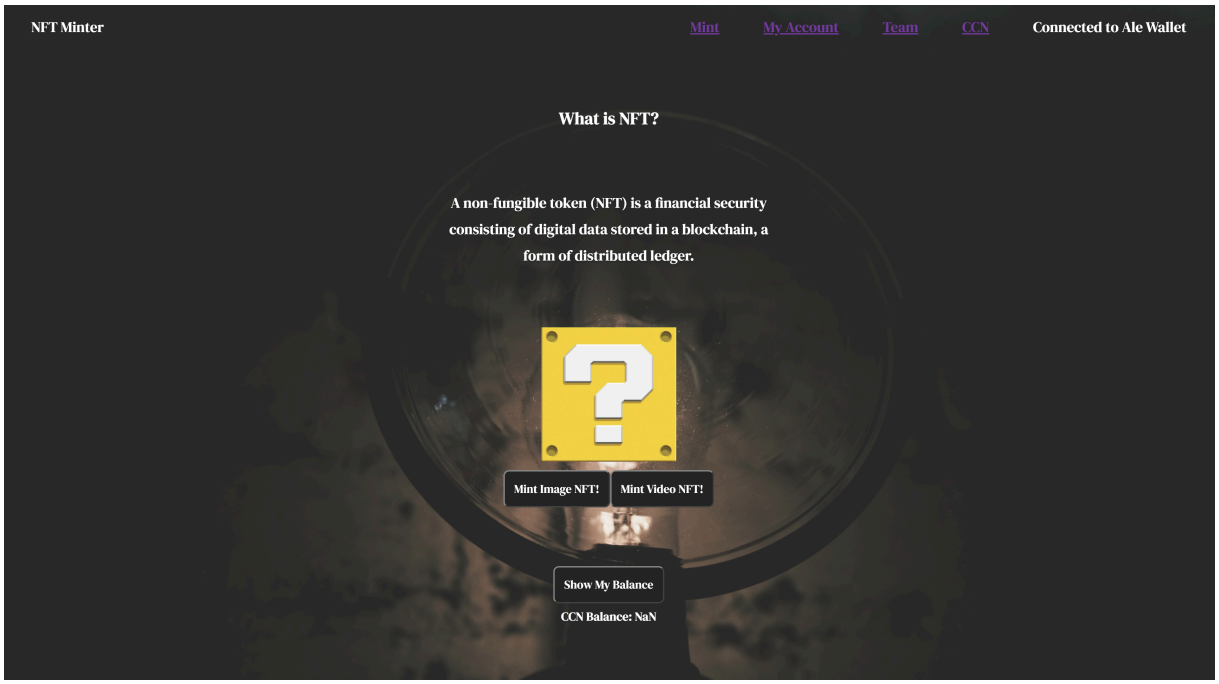

Figure 2: Page before Minting NFT.

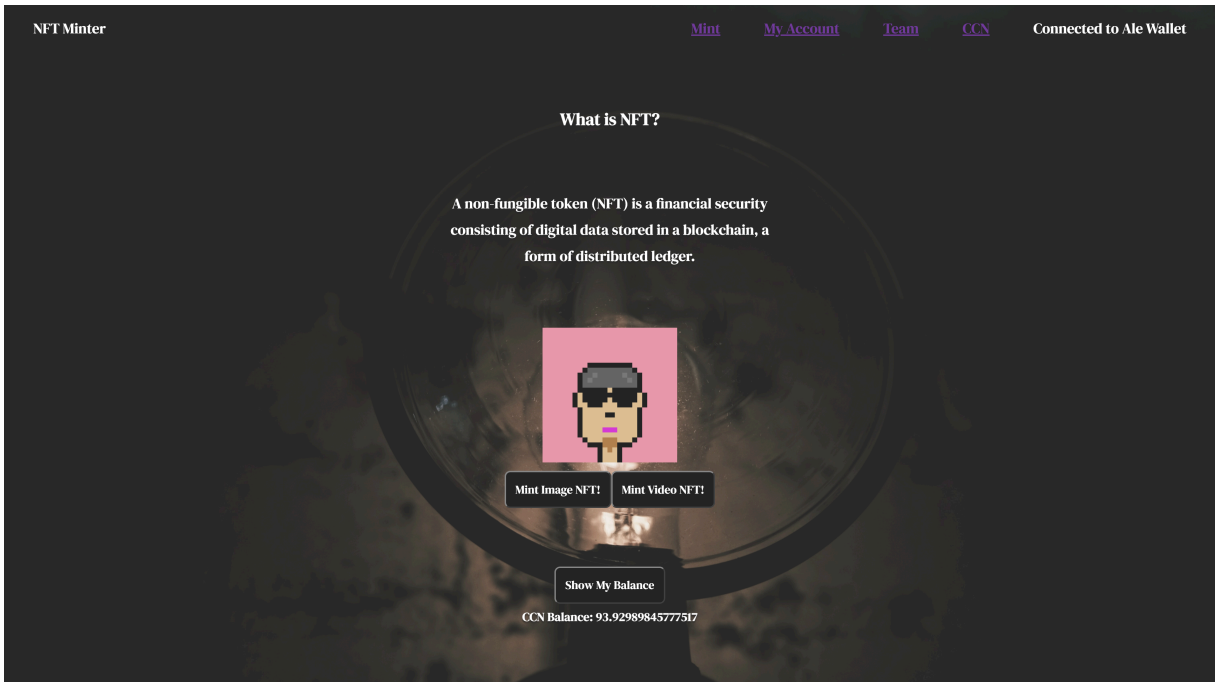

Figure 3: Page after Minting NFT.

2.2 Github

We have uploaded our code and img assests to Github, please visit https://github.com/xinhuang09/NFT-Minter

2.3 Video Demo

We have recorded a video to show how our DAPP works. Please visit https://youtu.be/VWwFHtxPnuI.

2.4 User Experience

Feedback from user: the use of this app is very convenient and simple. It is really novel that users can mint dynamic NFT videos. The style of the interface background and the pixel-style static NFT profile photo together provide users

with a mysterious and old-fashioned sense of feeling. To check the transaction records and account balance, users can simply click on the "My Account" or "Show Balance" button shown on the website.

3 Functionality

In this part we will introduce the methods we have used, we divided our whole project into three part called NFT Generation, Front End and Back End. For NFT Generation Part, we set a function to random splice different part of layers in the image. For our front end, we designed a website to achieve functions including connecting to Ale Wallet, transaction, image generation, balance and other information. For our back end, we defined several contracts on the concreate detail of rules about transaction.

For our project, we used Hygens as our testnet. The specific process is: Once the user has connected to our webpage and made a effective mint operation, an NFT image or video will be generated randomly and stored in IPFS(InterPlanetary File System) in the form of URL. After the transaction is finished, this URL will be deployed in the testnet.

3.1 NFT Generation

As we mentioned before, NFT acts as a non-duplicable digital certificate of ownership for any assigned digital asset, which means, it is impossible for two different user to have the same NFT image. To guarantee that principle, it is necessary that we have a large amount of NFT images. In order to achieve that, we have set different layers of image, and randomly combine them. The image below shows the process of generating an NFT image:

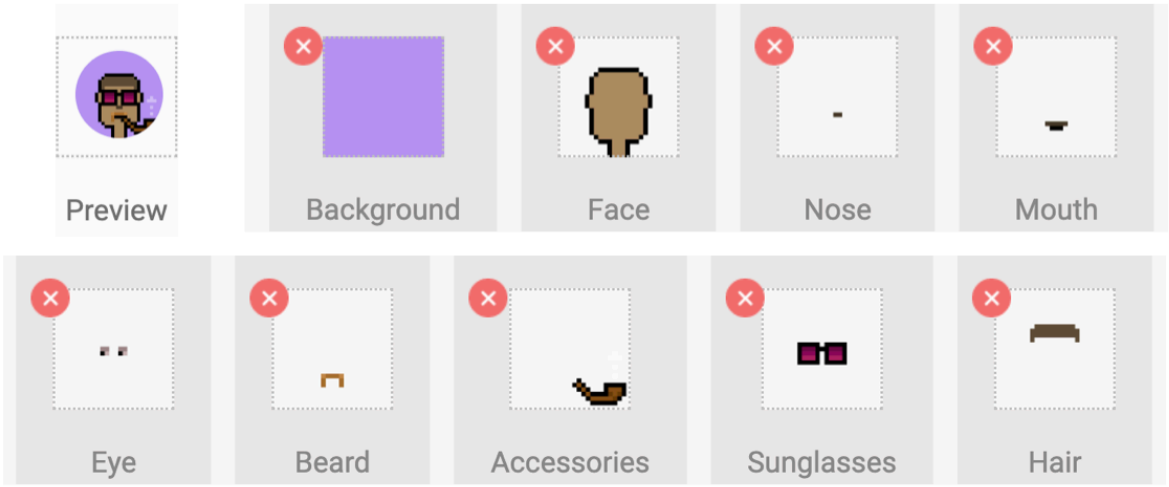

Figure 4: NFT production process.

We divided the image into 9 layers named background, face, nose, mouth, eye, bend, accessories, sunglasses and hair, each layer have 10 images, so after combining them randomly, there are enough images to mint.

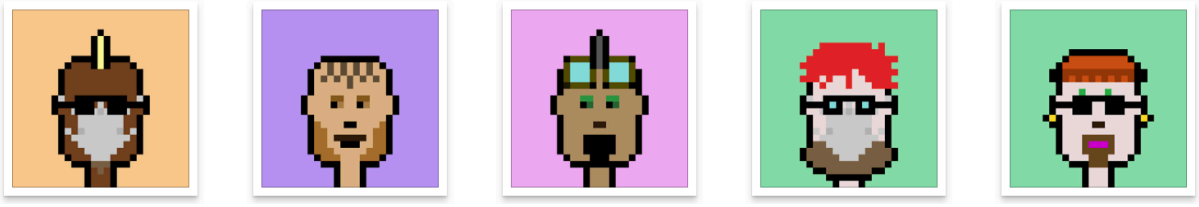

Figure 5: Examples of generated NFT.

Once the image is generated, we upload it into IPFS. IPFS (InterPlanetary File System) is a distributed system for storing and accessing files, websites, applications, and data. It is a public network, which means that anyone can start downloading and storing content on the network right away. The final image is stored in IPFS as a URL.

3.2 Front End

To make users easy to mint NFT images, we have created a web page in our front end. The web page is showed in figure 1 2 3. Here is the table of the functions of our page:

Table 1: Functions of the front-end page.

Button	Function
connect	Build a connection to the Ale Wallet account.
mint	Mint image (including image and video).
My Account	View account information, including transaction history, balance, etc.
Team	The link to our github webpage.
CCN	The link to CCN webpage.
Show Banlance	A shortcut for displaying account balance

As the table shows, there are 6 core buttons in our website, the first one is "connect" button, it is used for connecting to the account, once it is connected, the button will be turned into "connected to Ale Wallet", and there will be a square interface to show the NFT image if the user has minted one. The second one is called "mint", there are three "mint" buttons in our webpage, one is in the top right which is used for minting a NFT image, the other two in the center is used for minting a NFT or video image(we will introduce this function in detail in Section 4. Next is the "Account" button, after clicking this button, it will jump to another page which is also linked to the Ale Wallet account, and show some basic information about this user. Finally, as for the "Team" and "CCN" button, they are used for show some basic information about team members and CCN.

3.3 Back End

The main function of our back end is to set contracts about the detail of NFT transactions. Here is the table of our main functions in contracts:

Table 2: Descriptions of Functions in the back-end page.

Function	Description
IsPublicMintEnabled()	Check whether the account can mint or not.
BaseTokenUri()	Set the URI where the NFT images will be stored.
tokenURI()	This is existed in ERC721, as the base of BaseTokenURL function.
withdraw()	It is necessary to define that only owner can do this operation.
mint()	The most important function in our contracts, it defines the concreate detail of NFT transaction.
deposit_to_web()	It is also applicable to deposit money to the webpage.

It is noticeable that the most important function in our back end contracts is called "mint()", it defines the specific process of NFT transactions. Firstly, this is not applicable if the user is not connected. Secondly, there should be a "require" function to call "IsPublicMintEnabled", the purpose is to check whether it is allowed to mint. Next is checking if the mint value is correct, if it is wrong, the transaction should not happen (The true value equals to amount $\times$ mintprice). Finally, it is necessary to check whether the amount that the user want to pay exceeds the maximum in wallet, if does, the transaction should not happen.

4 Novelty

In our project, we made a DAPP for users to mint NFT, both static and dynamic NFTs. The principle of how to make an NFT is not complicated, but if the token we expect is not just a picture, but a gif or a complete video, how do we encrypt it so that it can be non-fungible?

For traditional approach to make image non-fungible, basically, we have:

$$O = \mathbf{F}(I) \tag{1}$$

where I stands for the input and O stands for the output, and **F** is the encryption function normally, it is a hash function, in blockchain.

By using equation (1), we can have a single pair of input-output relationship, it can encrypt an image, and if someone broadcast his info and the id (O) to the blockchain, an NFT is created. But there might be a question, if we want to encrypt a Gif or a Video, what should we do? There are many ways to do so, but in this project, inspired by the idea of Merkle Tree we learned in class, we propose a new approach to do so.

Recall the knowledge of Merkle Tree, it has an important property of tamper-evident, which means if one element in the tree has been changed, then the output of the Merkle Tree will be changed accordingly, as shown in figure 6 below.

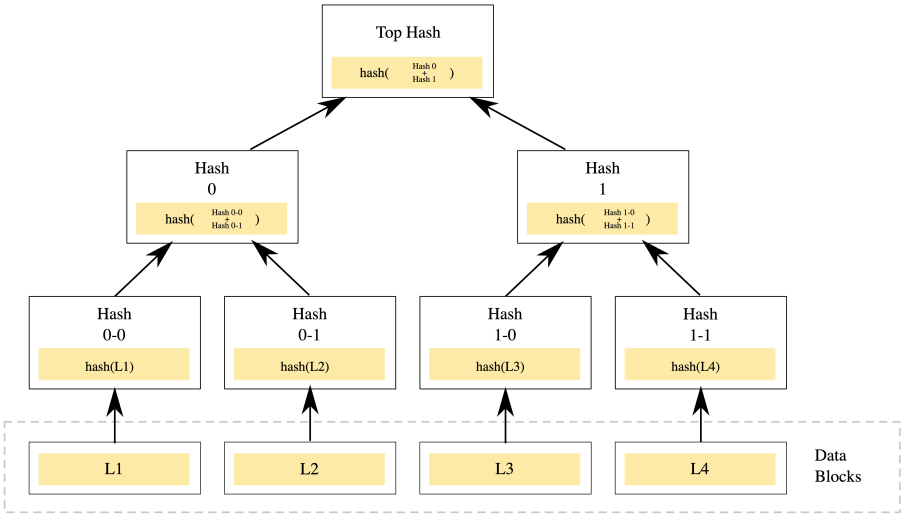

Figure 6: Merkle Tree (image source: Wikipedia - Merkle Tree).

By using the idea of Merkle Tree [4], we may treat every frame in the gif or video as an element in the data blocks, thus, we main combine those frames to the top hash, and take the top hash as the output of the encryption for the gif or the video. And the single pair input-output relation will become:

$$O_{Merkle} = \mathbf{F}(X_1^{\log k-1}, X_2^{\log k-1}) \tag{2}$$

and the input to the data blocks is $X = [X_1^1, X_2^1, \cdots, X_k^1]$, where there are k frames in total.

The final information we are going to send to the blockchain can be descripted as:

$$O = (O_{external}, O_{Merkle}) \tag{3}$$

where $O_{external}$ is encrypted metadata of the file and O_{Merkle} is the top hash of the Merkle Tree we have discussed above.

If we extend this concept, we can use this method to create NFTs for any file with more than one element composition, such as the video mentioned in this article (see it as a multi-frame picture).

5 Conclusion

In this project, we built a blockchain-based full-stack DAPP, and we conceived and programmed a smart contract to support this functionality. At the same time, we built a clean, beautiful front-end website to showcase this functionality.

In our DAPP, users can connect their own blockchain wallets and perform NFT minting. In addition to static NFT, we also provide dynamic NFT minting (based on the encryption method we mentioned in section 4), therefore, our DAPP is quite different from the common NFT Minting applications now. We will continue to try to make NFTs of different types of files in the future, so that more types of files or items have the possibility of being digitized.

Through this project, that is, the construction and programming of this full-stack blockchain DAPP, we have used the knowledge learned in the course, and have a deeper understanding of concepts such as smart contract,

NFT, merkle tree, hash encryption, etc. In this process, we systematically learned the solidity language, and also perceived the deterministic characteristics of solidity.

In short, in this project, we combine the theory and practice learned in the classroom, and fully understand the development and critical concepts in blockchain technology.

Acknowledgement

Here, we appreciate Professor Chong Li and Professor Chonggang Wang for their teaching and guidance in this semester, and we would like to thank the staffs on discord for their help in building this DAPP.

References

[1] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, pp. 99–111, 1991.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.

[3] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013.

[4] R. Merkle, "Protocols for public key cryptosystems," pp. 122–134, 04 1980.

A Github Repo

Our code is uploaded to Github Repo, you may visit https://github.com/xinhuang09/NFT-Minter.

B Authors

Table 3: Team Member of this Project.

Name	UNI	Email
Xin Huang	xh2510	xh2510@columbia.edu
Ziqi Zhang	zz2881	zz2881@columbia.edu
Weicong Zhang	wz2582	wz2582@columbia.edu
Zhengyu He	zh2480	zh2480@columbia.edu
Kairui Yin	ky2483	ky2483@columbia.edu