

CHRIST
(DEEMED TO BE UNIVERSITY)
B A N G A L O R E • I N D I A

MINTTOSS

by

Lakshmi Narayana B (1941029)

Anjana M S (1941053)

Under the Guidance of

Dr. Arokia Paul Rajan

A Project report submitted in partial fulfillment of
the requirements for the award of degree of
Bachelor of Computer Applications of
CHRIST (Deemed to be University)

April - 2024



CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE • INDIA

CERTIFICATE

*This is to certify that the report titled **MintToss** is a bona fide record of work done by **Lakshmi Narayana B (1941029)** and **Anjana MS (1941053)** of CHRIST (Deemed to be University), Bangalore, in partial fulfillment of the requirements of VI Semester BCA during the year 2024.*

Project In-charge

Head of the Department

Valued-by:

Name : Lakshmi Narayana B
Anjana MS

1.

Register Numbers : 1941029
1941053

2.

Examination center : CHRIST (Deemed to
be University)

Date

:

ACKNOWLEDGMENTS

The project, MintToss – an NFT Marketplace, was successfully completed with the assistance of many individuals. Firstly, we express our sincere gratitude to our Vice-Chancellor, Dr. Fr. Joseph CC, Pro Vice-Chancellor, Dr. Fr. Viju P D, Head of the Department, Dr. Ashok Immanuel V, Coordinator, Dr. Sagaya aurelia and the faculty for providing us with this invaluable opportunity to put in and improve our technical skills and learn new technologies and work on a practical idea.

We express our gratitude to our project guide **XXX Guide XXX** who took a keen interest in helping and guiding us throughout our project work by providing all the necessary information for coming up with a good result.

We would also like to thank other teachers for the suggestions, continuous feedback, and supervision of our project work. We are thankful and fortunate enough to get constant encouragement, support, and guidance from the faculty of BCA, and our fellow batchmates who helped us in successfully completing our project work.

ABSTRACT

MintToss is an NFT (Non-Fungible Token) Marketplace that facilitates NFT trading. An NFT, or non-fungible token, is a unique token on a blockchain that is probably unique. This site is a peer-to-peer marketplace for NFTs, rare digital items and crypto collectibles. One can buy, sell and create an NFT using this site. It's a platform that enables the user to preserve and trade NFT or Non-Fungible Tokens easily. The user can usually buy or auction these tokens at a fixed rate. The user must have a wallet with enough Ethereum currency as balance to make transactions and store the tokens to use this marketplace.

To use this marketplace, the user should have a Metamask account and must connect the account to the marketplace. Only then the user can do the following:

- Create an NFT and define all preferred parameters.
- List your digital goods for sale and wait for the completion of the moderation.
- Explore and buy NFTs from the marketplace.
- View the transactions.
- View the ranking of the authors of the NFTs.

MintToss utilizes particular transaction protocols known as smart contracts. These protocols regulate the connections between the seller and the buyer. Moreover, these smart contracts contain identifying data associated with an NFT. Hence, the process of buying and selling tokens becomes user-friendly and convenient for the users.

TABLE OF CONTENTS

Acknowledgments	iii
Abstract	iv
List of Tables	vii
List of Figures	viii
1. Introduction	1
1.1 Background of the project	1
1.2 Objectives	1
1.3 Purpose, Scope and Applicability	1
1.4 Modules	2
1.5 Major outcome	3
1.6 Highlight of the Project	3
1.7 Tools used	3
2. System Analysis and Requirements	4
2.1 Existing System	4
2.2 Limitations of Existing System	4
2.3 Proposed System	4
2.4 Benefits of The Proposed System	5
2.5 Features of The Proposed System	5
2.6 System requirement specifications	
2.6.1 User Characteristics	6
2.6.2 Software and Hardware requirements	6
2.6.3 Constraints	7
2.6.4 Functional Requirements	7
2.6.5 Non-Functional requirements	9
2.7 Block diagram	11
3. System Design	12
3.1 System Architecture	12
3.2 Module design	13
3.3 Data flow diagram	14
3.3.1 DFD Level-1	14

3.3.2 DFD Level-2	15
3.4 ER Diagram	17
3.5 Database Design	18
3.5.1 Table design	18
3.5.2 Data integrity and constraints	21
3.5.3 Data Dictionary	22
3.6 Interface and procedural design	23
3.7 Reports Design	25
4. Implementation	26
4.1 Coding standards	26
4.2 Coding Details	27
4.3 Screenshots	36
5. Testing	45
5.1 Testing Strategies	45
5.1.1 System Testing	45
5.1.2 Test data Implementation	45
5.1.3 Test Characteristics	46
5.1.4 Black box testing	46
5.1.5 White box testing	47
5.2 Test Cases	49
5.3 Test Reports	50
6. Conclusion	51
6.1 Design and Implementation Issues	51
6.1.1 Design Issues	51
6.1.2 Implementation Issues	51
6.2 Advantages and Limitations	51
6.2.1 Advantages	51
6.2.2 Limitations	52
6.3 Future and Scope of the Project	52
References	53

LIST OF TABLES

Table No.	Table Name	Page No.
2.1	Functional Requirements	7
2.2	Non-functional requirements	9
3.1	User table	18
3.2	User's NFT details	18
3.3	NFT details	19
3.4	Rankings table	19
3.5	Assets table	20
3.6	Sales table	20
3.7	User table	22
3.8	User's NFT details	22
3.9	NFT details	22
3.10	Rankings table	22
3.11	Assets table	22
3.12	Sales table	23
5.1	Test Cases	49
5.2	Test Reports	50

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
2.1	Block diagram	11
3.1	System Architecture	12
3.2	Client-side flow	12
3.3	DFD level-1 diagram	14
3.4	DFD level-2 diagram	15
3.5	Blockchain representation	16
3.6	ER Diagram	17
3.7	Home page	23
3.8	Collections page	24
3.9	Search module	24
3.10	Transactions page	25
3.11	Transactions report	25
4.1	Metamask Authentication	36
4.2	Metamask Login	37
4.3	Home page – Browse by category	37
4.4	Home page – New Items	38

4.5	Home page – Top sellers	38
4.6	Home page – How it works?	39
4.7	Mint NFT Form	39
4.8	Minting NFT with preview	40
4.9	NFT Description	40
4.10	Minted NFT – Ready	41
4.11	NFT listed on new items	41
4.12	Listing NFT out on the marketplace	42
4.13	Listed NFTs	42
4.14	My Assets	43
4.15	Search page	43
4.16	Contact Us page	44
4.17	Ranking Authors page	44
5.1	Flow graph	46

1. INTRODUCTION

1.1 BACKGROUND OF THE PROJECT

MintToss is an NFT (Non-Fungible Token) Marketplace that facilitates NFT trading. An NFT, or non-fungible token, is a unique token on a blockchain that is probably unique. This site is a peer-to-peer marketplace for NFTs, rare digital items, and crypto collectibles. One can buy, sell and create an NFT using this site. It is a platform that enables you to preserve and trade NFT or Non-Fungible Tokens easily. The user can usually buy or auction these tokens at a fixed rate. The user must have a wallet in order to make transactions and store your tokens to use this marketplace. This marketplace also provides the ranking of the authors.

1.2 OBJECTIVE

To create a peer-to-peer web3 service that facilitates users to create, buy and sell NFTs available on public blockchains.

1.3 PURPOSE, SCOPE AND APPLICABILITY

Purpose

- An NFT marketplace that supports all the types of tokens such as videos, illustrations, photographs, music, collectibles and much more and to avail the user to mint his or her own NFTs and to buy and sell.
- This website encourages users to buy, sell and trade the current trend of investments that is NFTs.

Scope

- Anyone who has an Ethereum based crypto wallet like Metamask can buy and sell NFTs through this marketplace.
- It is now limited as a web application.
- Only browsers with Metamask wallet Extension will be able to view and avail the facilities of this site.

Applicability

- The users can add NFTs to different categories such as music, collectibles, trading cards and more.
- The users can trade NFTs using Ethereum.

1.4 MODULES

Explore feed: In this module all the available NFTs accounts will be shown and the user can select any NFT to buy or can just view the portfolios.

Ranking of NFTs: In this module, the most viewed and purchased NFTs will be listed along with their statistics like volume, floor price, owners, items etc.

Buying: One of the main functionalities in this site is the buying option where the users can buy their desired NFTs by paying the value quoted for that particular NFT.

Creating: The users can create their own NFT, such as an illustration, photograph, 2D or 3D design, and so on. Anything can be created and posted on your profile. You can set a price for your art and it is ready to be sold.

Selling: This option lets the user sell the NFTs that they created or owed. There will be bidding for your item and the highest bid will get the item.

Wallet: The list of available modes of payment options are shown, either cash or blockchain based currencies. The user can add value to his/her account's wallet and purchase the NFTs.

Profile: The Profile section will show the profile of the user. All the details related to his/her NFTs, purchases, sales, and wallet details will be displayed.

Settings: In the settings section the user can make settings related to his profile.

Filter Tab: In the filter option the user can filter the search result, depending on his/her choice of the NFTs, this will directly take the users to their preferred NFT clubs where their collections are posted.

Help: In this module, we will be providing the basic knowledge related to different options available on the site for beginners. All the FAQs will be posted for the users to refer to for any of their doubts. The user can also post new questions which will be answered. Links to our communities like Email, Facebook, Instagram and Twitter will also be available.

1.5 MAJOR OUTCOME

An aim to build an open digital economy NFTs have exciting new properties: they're unique, probably scarce, tradeable, and usable across multiple applications. Just like physical goods, you can do whatever you want with them! You could throw them in the trash, gift them to a friend across the world, or go sell them on an open marketplace. But unlike physical goods, they're armed with all the programmability of digital goods.

1.6 HIGHLIGHT OF THE PROJECT

- The user can mint, buy and sell NFTs using this marketplace.
- User-friendly interface.
- Ranking of authors of the NFTs are displayed.
- Various sub-headings like Marketplace's choice, New Items, Browse by Category are shown for the users to easily categorize listed NFTs
- NFTs are categorized into various domains like art, music, trading cards, collectibles and so on.

1.7 TOOLS USED

- JavaScript
- NodeJS
- Metamask
- Hardhat
- Openzeppelin
- Infura
- VS Code
- Git

2. SYSTEM ANALYSIS AND REQUIREMENTS

This chapter describes the system requirements and its analysis. It includes hardware and software requirements as well as the functional and non-functional requirements.

2.1 EXISTING SYSTEM

A good example of an existing system is Sorare, which specializes in selling and buying football players' trading cards through the form of NFT. Sorare is a fantasy game of football, where players buy, sell, trade, and manage a virtual team with digital player cards. The game uses blockchain technology based on Ethereum and was developed in 2018 by Nicolas Julia and Adrien Montfort. Since, its target audience consists of football lovers from all over the world, it encountered huge success and immense traffic among the football fans. In Sorare marketplace, the users can collect the cards of the players. These cards will act as a representation of the players, which has the unique moves and other states of the players. These collectibles can be considered as the key for playing the NFT based fantasy game hosted by the marketplace. It can also be considered to be an NFT asset that has high value.

2.2 LIMITATIONS OF EXISTING SYSTEM

The existing systems mostly provide only services for NFTs of specific domains with target audiences. So, there are not many general systems that provide all the services. Most of the existing systems don't provide the ranks of the NFTs based on the necessary statistics, and also few sites don't allow the users to mint NFTs.

2.3 PROPOSED SYSTEM

MintToss is an NFT (Non-Fungible Token) Marketplace that facilitates NFT trading. An NFT, or non-fungible token, is a unique token on a blockchain that is provably unique. This site is a peer-to-peer marketplace for NFTs, rare digital items and crypto collectibles. One can buy, sell and create an NFT using this site. It is a platform that enables you to preserve and trade NFT or Non-Fungible Tokens easily. You can usually buy or auction these tokens at a fixed rate. You must have a wallet to make transactions and store your tokens to use this marketplace. This marketplace also ranks the authors of the NFTs based on their trading history.

The features of this marketplace are listed below:

- Auto detecting if the user has a Metamask extension.
- Auto connect the application with the user's Metamask wallet.
- Detects the connected network as well as account changing.
- Create & Sell and Buy NFTs.
- Calculate the user funds.
- Give the user the ability to collect his or her funds.
- Calculate the Top Sellers as well as their selling rank.
- Sync the marketplace user avatar with the Metamask avatar.
- The user can view and track all the assets he or she owns in a single page.
- Track your transactions on Ether Scan.
- Filter NFTs according to their category.
- Working Contact form.
- During minting, the user can preview how the item looks like on the marketplace before he or she uploads it to the blockchain.

2.4 BENEFITS OF THE PROPOSED SYSTEM

Since most of the existing systems only provide NFTs of specific domains with target audiences, we extend our marketplace for the general audience to buy, sell and create any kind of NFTs. The user can create NFTs under different categories such as music, collectibles, trading cards and so on. The authors will be ranked based on the highest trading value of their NFTs. So, MintToss will be a marketplace where the user can easily buy, sell and create NFTs by paying a small gas fee.

2.5 FEATURES OF THE PROPOSED SYSTEM

- Liquidity - Because the NFT marketplace allows for trading, the tokens are extremely liquid. The liquidity pool on the marketplace is accessible to token users.
- User friendly - The NFT marketplace website is simple to use, making it a welcoming environment for beginners. It also provides support to the new users on how to use the marketplace.

- Crypto wallet Integration - Because NFT payments are now made through cryptocurrencies, the NFT marketplace should include a safe and reliable crypto wallet integration mechanism.
- Security - The smart contract and decentralized platform ensure that users' transactions and related information are completely safe and secure.
- During minting, the user can preview how the item looks like on the marketplace before he or she uploads it to the blockchain.
- Auto connect the application with the user's Metamask wallet.
- Auto detecting if the user has a Metamask extension.
- Calculate the Top Sellers as well as their selling rank.
- The user can view and track all the assets he or she owns in a single page.
- Filter NFTs according to their category.

2.6 SYSTEM REQUIREMENTS SPECIFICATION

2.6.1 User Characteristics

Any individual who wants to make digital investments using Ethereum cryptocurrency can make use of this site. This will also help talented people to sell their artwork and make money. People who are interested in art works can buy them and keep them in their gallery or can further trade them according to their wish. A user can create and sell NFTs as different categories such as music, collectible, trading cards, art and so on.

2.6.2 SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements

- JavaScript
- NodeJS
- Metamask
- Hardhat
- Openzeppelin
- Infura
- VS Code
- Git

Hardware Requirements

- Processor - 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set
- Graphic card - Not required
- Disk Capacity - 1-2 Gb of Disk Capacity
- RAM - 2GB of Memory

2.6.3 CONSTRAINTS

Our proposed system MintToss is a web3 system that is implemented using Next.js by integrating it with the Polygon framework for building and connecting Ethereum-compatible blockchain networks. Hardhat is a development environment that is used by us to compile, deploy, test and debug our Ethereum software. Truffle and Ganache frameworks are used to create a deterministic environment. For a user to view and access the site, the user must have a Metamask wallet extension in his or her browser. The user should create an account in the Metamask wallet to connect his or her wallet to the site. This wallet is used to make payments while buying or minting an NFT. So, a user must have a Metamask account to access all the services of this site.

2.6.4 FUNCTIONAL REQUIREMENTS

Table 2.1 Functional Requirements

Requirement Id	Requirement	Description
M1_FR1	Wallet Extension	In this module, the user's browser is verified if it has a Metamask extension or not
M1_FR2	Connect wallet	The user must connect their wallet to the marketplace in order to avail the services of the site.
M2_FR1	Explore Feed	In this module, all the available NFTs accounts will be shown and the user can select any NFT to buy or can just view the portfolios.

M2_FR2	Ranking of NFTs	In this module, the most viewed and purchased NFTs will be listed along with their statistics like volume, floor price, owners, items etc.
M2_FR3	Filter Tab	In the filter option the user can filter the search result, depending on his/her choice of the NFTs, this will directly take the users to their preferred NFT clubs where their collections are posted.
M3_FR1	Buying	One of the main functionalities in this site is the buying option where the users can buy their desired NFTs by paying the value quoted for that particular NFT.
M4_FR1	Creating	The users can create their own NFT, such as an illustration, photograph, 2D or 3D design, and so on. Anything can be created and posted on your profile. You can set a price for your art and it is ready to be sold.
M4_FR2	Selling	This option lets the user sell the NFTs that they created or owned.
M4_FR3	Listing	<p>The user can list the NFT he or she owns for further trading. Only when a NFT is listed by the owner will another user be able to buy it.</p> <p>The owner can put up a different offer price while listing it on the marketplace.</p>
M5_FR1	Balance	The user must have enough Ethereum to mint, buy and list NFTs.

M6_FR1	Profile	The Profile section will show the profile of the user. All the details related to his/her NFTs, purchases, sales and wallet details will be displayed.
M6_FR2	Profile Settings	In the settings section the user can make settings related to his profile.
M7_FR1	Contact Us	In this module we will be providing the basic knowledge related to different options available on the site for beginners. All the FAQs will be posted for the users to refer for any of their doubts. The user can also post new questions which will be answered. Links to our communities like Email, Facebook, Instagram and Twitter will also be available.

2.6.5 NON-FUNCTIONAL REQUIREMENTS

Table 2.2 Non-Functional Requirements

Requirement ID	Requirement	Description
NF_R1	Performance	<ul style="list-style-type: none">● As it's a web site, the network, hardware and other related infrastructure plays a vital role in determining the application performance.● Data compression approach has been applied to reduce the network burden.● Number of navigations and touch are reduced to the minimum.● Basic and simple color and graphics settings are implemented for high

		performance.
NF_R2	Safety/Security	<ul style="list-style-type: none"> ● It is the state of being "safe", the condition of being protected against physical, social, spiritual, financial, political, emotional, occupational, psychological, educational or other types of consequences of failure, damage, error, accidents, harm or any other event which could be considered non-desirable. ● This can take the form of being protected from the event or from exposure to something that causes health or economical losses. It can include protection of people or of possessions.
NF_R3	Quality requirements	<p>1. Reliability</p> <ul style="list-style-type: none"> ● The system provides storage of all databases on redundant computers with automatic switchover. ● The reliability of the overall program depends on the reliability of the separate components. ● The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. <p>2. Maintenance</p> <ul style="list-style-type: none"> ● A commercial database is used for maintaining the database and the application server takes care of the site. ● Also, the software design is being done

		<p>with modularity in mind so that maintenance can be done efficiently.</p> <p>3. Usability</p> <ul style="list-style-type: none"> As the users of the system requirements are different for Everyone, the level of functionalities should require a better understanding of the system. The interface for each type of user kept very simple and complete for ease of use. Manuals, demos or the documents made available, simple, clear and definite set of interfaces makes the context easy to understand and use.
--	--	---

2.7 BLOCK DIAGRAM

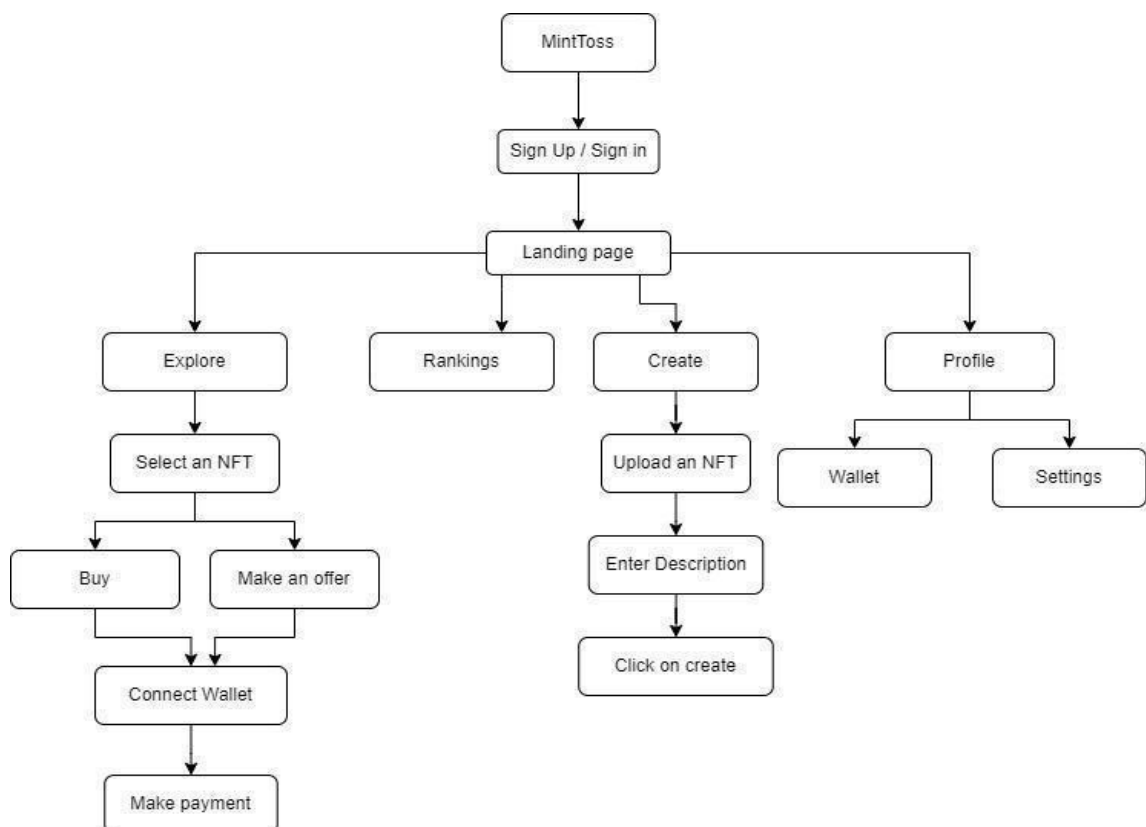


Fig. 2.1 Block Diagram

3. SYSTEM DESIGN

This Document shows the list of Modules that are there on the MintToss NFT marketplace website along with the System Architecture. The Data Flow Diagram and the ER Diagram are also mentioned so that the application of the site can be easily understood. The Sample UI Screen and the tables in the database have been listed out.

3.1 SYSTEM ARCHITECTURE

MintToss is a web application that allows users to buy and sell NFTs. This system was created in order to overcome the limitations of the existing systems. The user can categorize their NFTs into various domains. So, all the data is stored, retrieved and can be updated with the help of a database. Since this system is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms, it is a three-tier architecture.

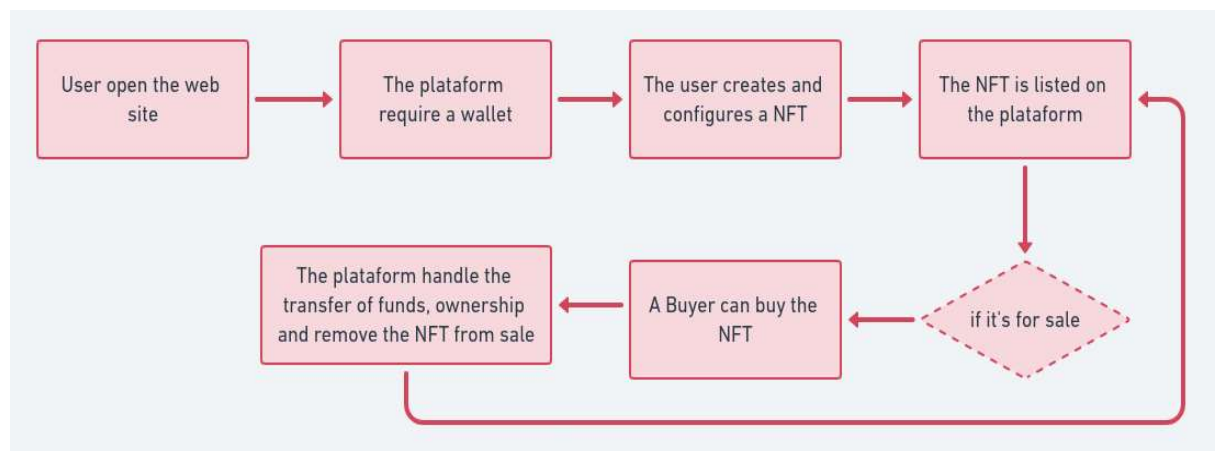


Fig 3.1 System Architecture



Fig 3.2 Client-side flow

3.2 MODULE DESIGN

- **Home** - It is the primary feature. It should contain such info as bids, preview, owners, price history, and more.
- **Advanced token search** - A customer must get robust data on items they need quickly, with minimum effort. An NFT marketplace should have all items sorted by some features (for example, music, images, videos, art, memes). Quick search enhances customer satisfaction.
- **Filter** - This functionality is similar to the previous one as the main idea is helping choose the right product fast and effortlessly. Divide all offers into several categories that impact buyer's decisions in most situations. Those can be prices, recent goods, hot offers, best-selling, and more. Users will pick items they need faster, and it increases the probability of buying them.
- **Creating Listings** - Giving a right to customers to develop and send collectibles. Ensuring they can do that quickly and with no obstacles. Generating a page where customers can submit a file, typing in the specific item data. Such info as title, tags, and description is a must.
- **Listing status** - Those who offer goods and pass item verification procedures should benefit from this option. It allows checking the status of the confirmation procedure. This feature is useful for implementing collectible verification.
- **Bidding options** - Making it possible to both purchase items and bid on them is critical for any e-commerce project. It attracts more users as some are interested in flexible pricing and do not wish to buy collectibles at their starting fees. Bidding is always fun. Do not forget to add an expiration date for an auction feature. Registered participants should have an opportunity to see information about the current status of their bids. It will help them decide whether to buy or keep on placing new bids. An auction watchlist is like a separate important feature.
- **Wallet** - Users need a safe place to receive and store their non-fungible tokens. Not all options are suitable as some of them may have certain threats to the security of funds. That is why the NFT market service must have an initially inserted wallet for saving and submitting tokens with no fear. Create and offer a connected, "native", wallet instead of forcing your buyers to sign up with other online wallets. Put their convenience in the first place. We might create a list of

the top-preferred wallets and add them to our platform. Doing everything to make storing, submitting, and obtaining tokens as simple as possible.

- **Ratings** - This characteristic is for newbies. Beginners may have no idea where to start from, how the system works, and how to choose items fast. Having a look at one's rating to find out whether other users consider this specific seller a reliable one might be enough. Thanks to ratings, the platform's participants can rate others and provide feedback based on their impressions. It will help others see how credible every user is.

3.3 DATA FLOW DIAGRAM

3.3.1 DFD Level 1

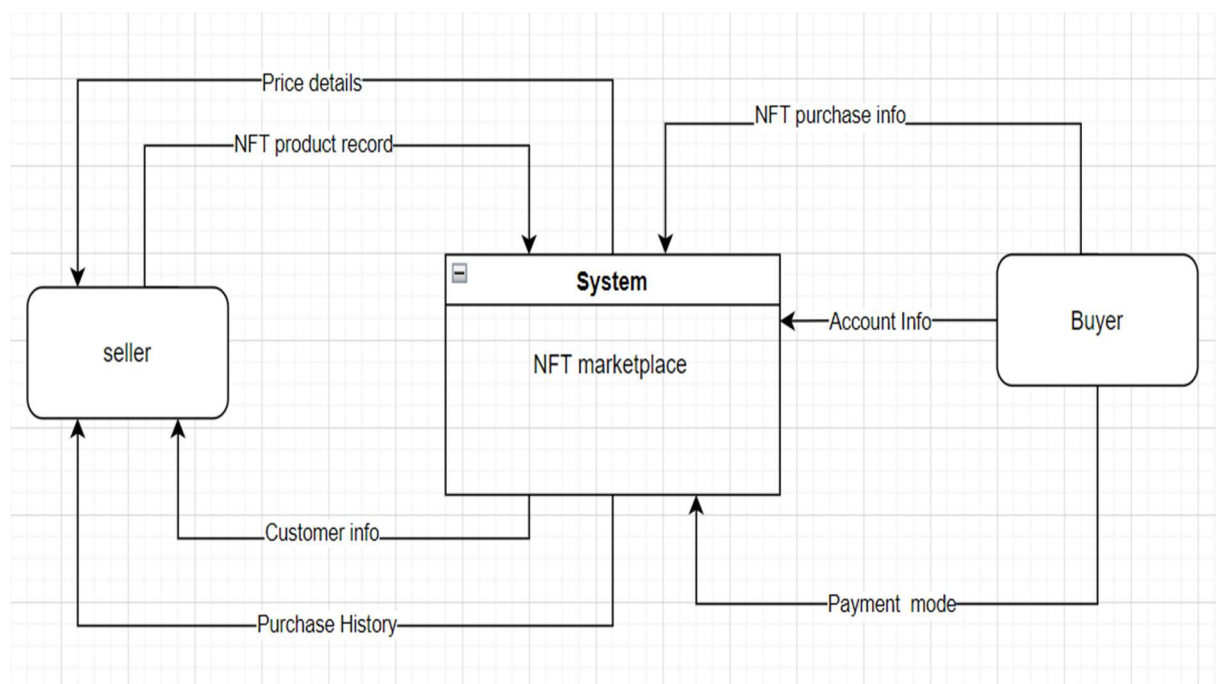


Fig 3.3 DFD level - 1 diagram

3.3.2 DFD Level 2

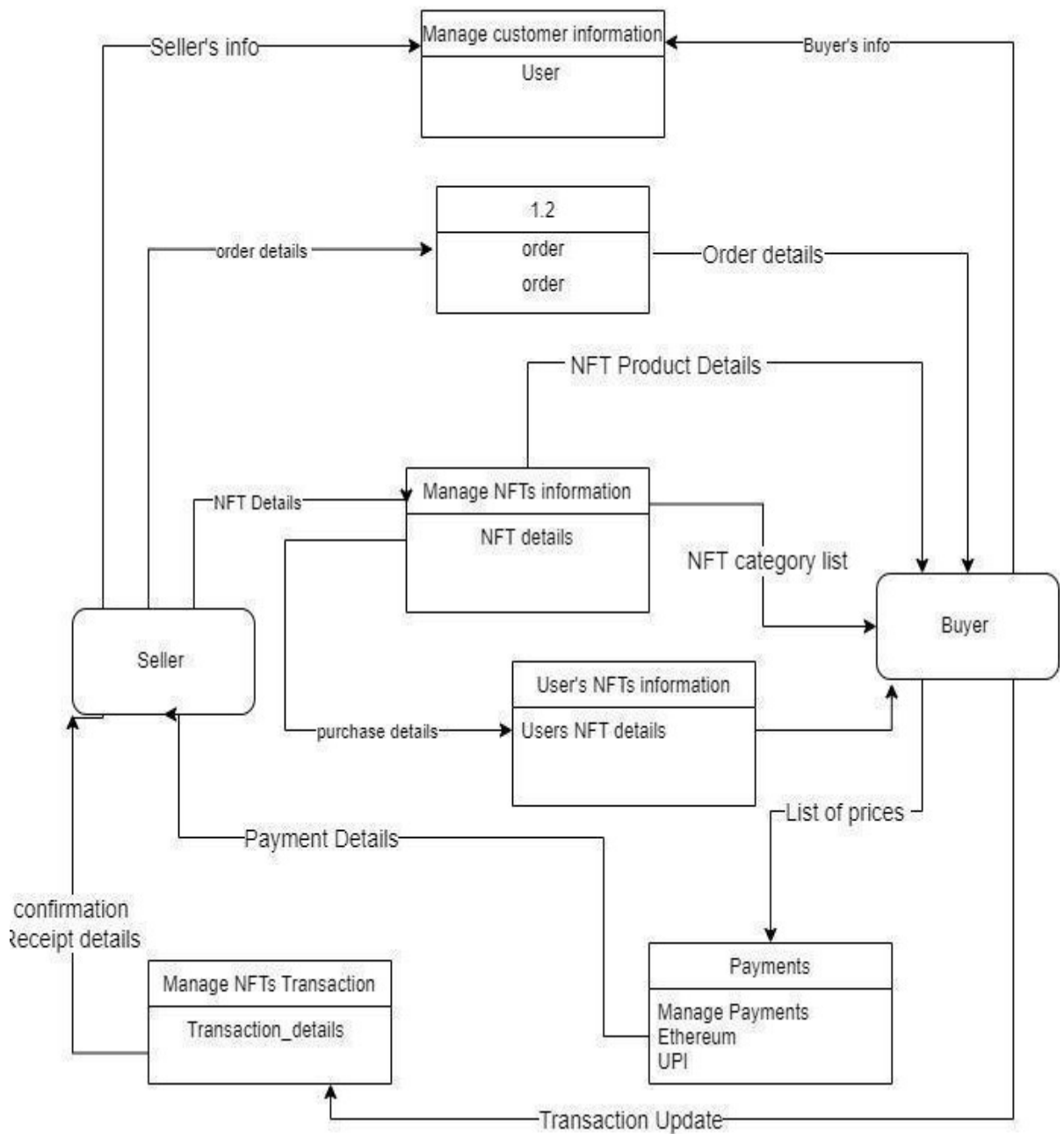


Fig 3.4 DFD level - 2 Diagram

Blockchain data process representation

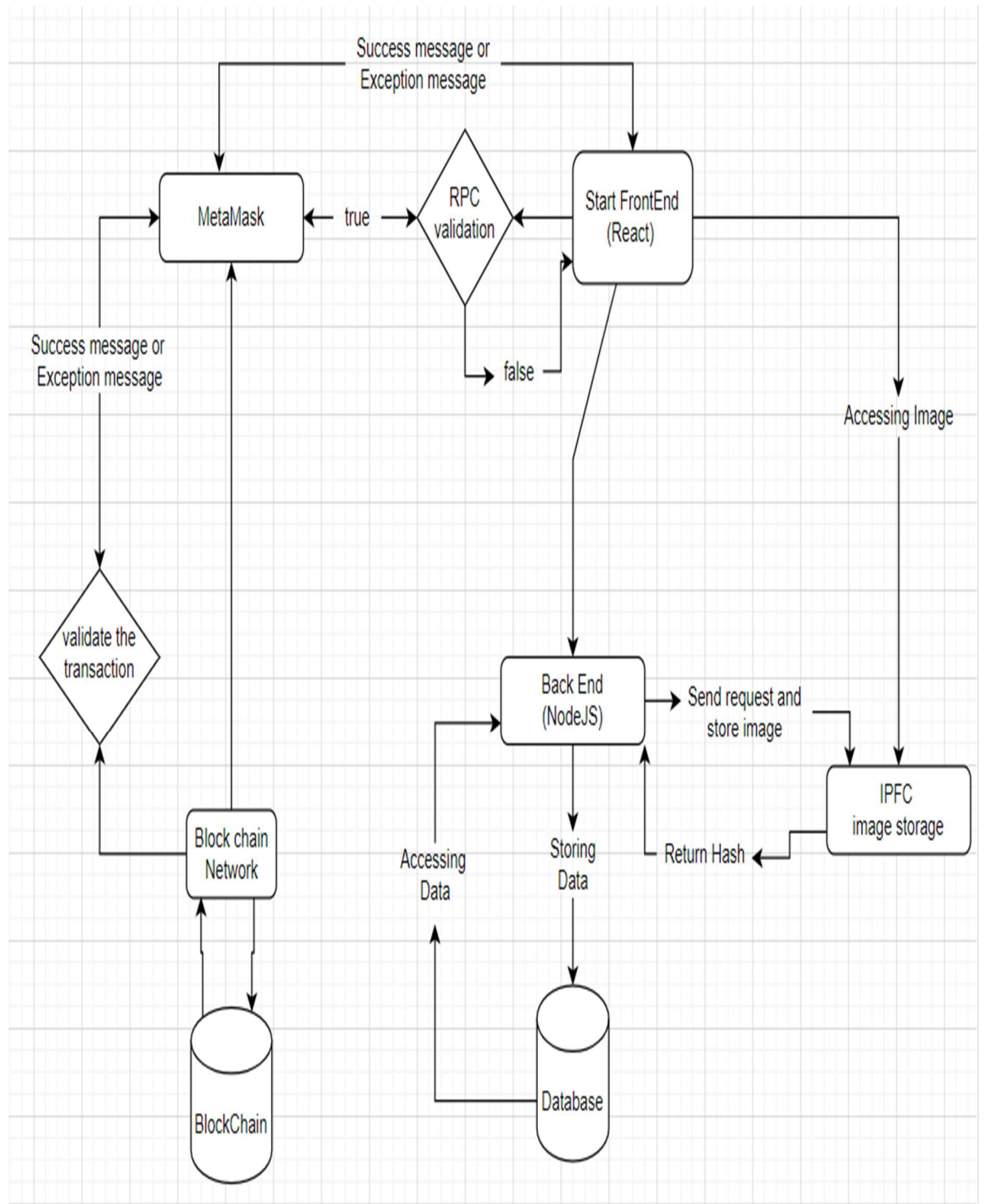


Fig 3.5 Blockchain representation

3.4 ER DIAGRAM

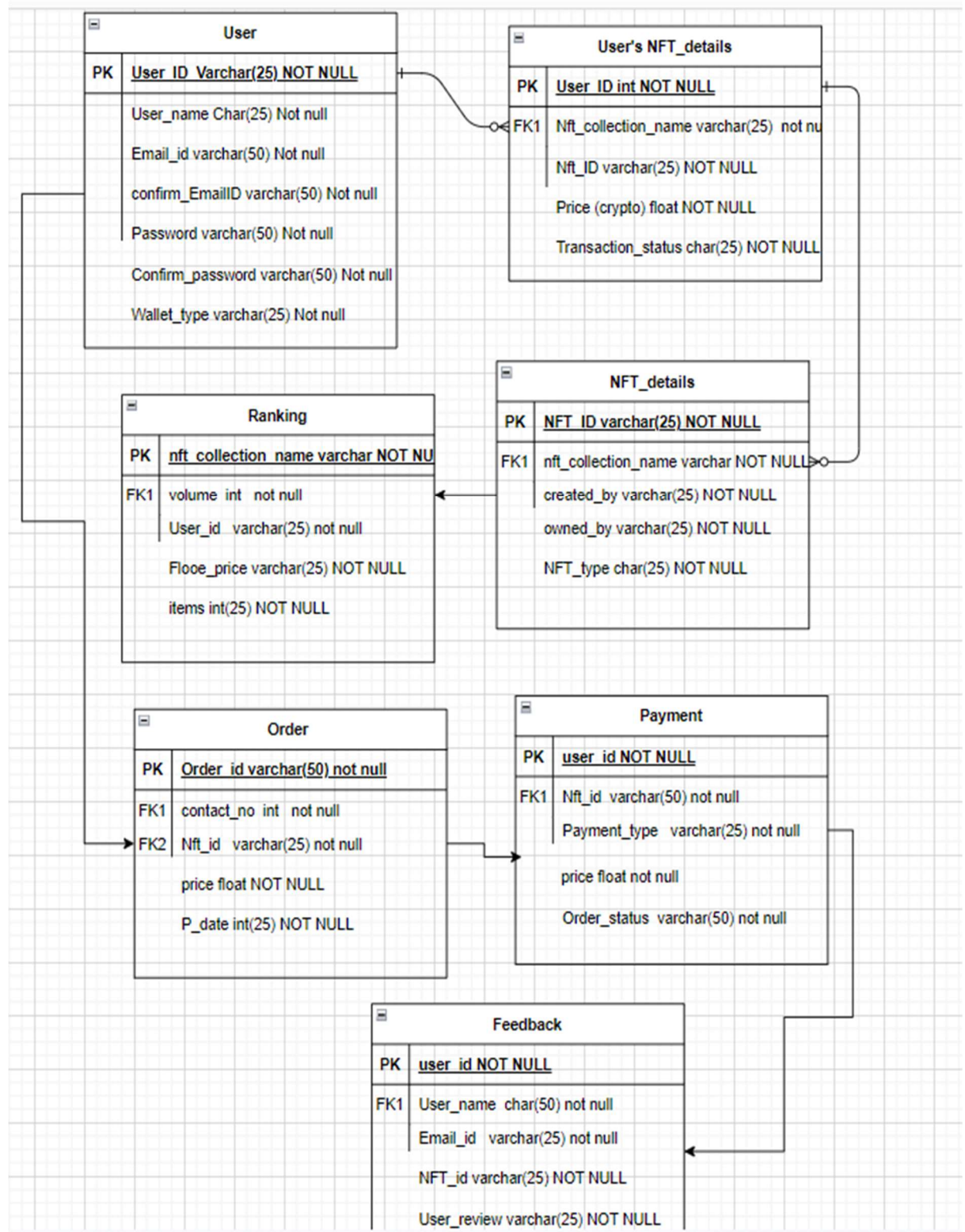


Fig 3.6 ER diagram

3.5 DATABASE DESIGN

3.5.1 Table Design

Table 3.1 User table

Sno.	Attributes	Data type	Description	Constraint
1	User_id	Varchar(25)	Wallet Id of the user.	PK
2	Name	Char(25)	Name of the user declared by Metamask wallet.	Not null
3	Connect_wallet	Boolean	Confirmation of the wallet connection.	Not null
4	Wallet_type	Char(25)	Type of wallet of the user	Not null

Table 3.2 User's NFT Details

Sno.	Attributes	Data type	Description	Constraint
1	User_id	Varchar(50)	Wallet Id of the user	PK, FK
2	Collection_name	Varchar(50)	Name of the NFT Collection	Not null
3	Nft_id	Varchar(50)	ID of the NFT	FK
4	Image_url	Varchar(100)	URL of the image	Not null, Unique
5	Price (crypto)	Float	Price of the NFT	Not null
6	Transaction_status	Char(25)	Status of the transaction of the NFT	Not null

Table 3.3 NFT Details

Sno.	Attributes	Data type	Description	Constraint
1	Nft_id	Varchar(50)	ID of the NFT	PK
2	Image_url	Varchar(100)	URL of the image	Not null, Unique
3	Collection_name	Varchar(50)	Name of the NFT Collection	Not null
4	Collection_id	Varchar(50)	ID of the collection	Not null
5	Author	Varchar(50)	User_ID of the creator of the NFT	Not null
6	Owned_by	Varchar(50)	User_ID of the owner of the NFT	Not null
7	Category	Char(25)	Type of the NFT	

Table 3.4 Rankings Table

Sno.	Attributes	Data type	Description	Constraint
1	Author	Varchar(50)	Id of the Author	PK
2	Volume	Int	Net trading worth	Not null
3	Floor_price	Float	Minimum amount of the NFT	Not null
4	Items	Int	No. of items in the collection	Not null

Table 3.5 Assets Table

Sno.	Attributes	Data type	Description	Constraint
1	Nft_id	Varchar(50)	ID of the NFT	FK
2	Collection_name	Varchar(50)	Name of the Collection	Not null
3	Collection_id	Varchar(50)	Id of the collection	Not null
4	Description	Varchar(150)	Description of the NFT	
5	Contract_date	Date time	Timestamp of the contract created	Not null
6	Image_url	Varchar(100)	URL of the image	Not null, Unique
7	Author	Varchar(50)	Id of the Author	FK

Table 3.6 Sales Table

Sno.	Attributes	Data type	Description	Constraint
1	Nft_id	Varchar(50)	Id of the NFT	FK
2	Collection_name	Varchar(50)	Name of the Collection	Not null
3	Collection_id	Varchar(50)	Id of the collection	Not null
4	Contract_address	Varchar(150)	Smart contract address	Not null
5	Contract_date	Date time	Timestamp of the contract created	Not null
6	Price (crypto)	Float	Price of the NFT	Not null
7	Seller_id	Varchar(50)	Id of the seller	Not null, unique
8	Buyer_id	Varchar(50)	Id of the buyer	Not null,

				unique
9	Image_url	Varchar(100)	URL of the image	Not null, Unique

3.5.2 Data Integrity and Constraints

Data integrity constraints refer to a basic set of rules in order to maintain the quality of the information that is added to the database. Many measures have been taken like the encryption of the backend process so that it is not visible to the moderator. Data that is processed within the web app has also gone through the process of Data Standardization which is the process of converting raw data into the correct format so that the data can be properly processed and analysed. Data Standardization is very crucial as it involves converting that data into a uniform format for making data relevant for examining. Data will also undergo Normalization principles which is basically the process of organizing data in order to avoid duplication of data in the database.

Since more focus has been provided to have easy navigation and easy understanding of the features which makes the usability of the web app more and more convenient and makes the web app easy to use.

1. Wallet (Metamask wallet connection)

- User should have a new Metamask account to avail of services of the website. Currencies are transferred using a Metamask account.
- Username – Declared by Metamask
- Users must have a certain amount of Ethereum to do the minting.

2. NFT Details

- Nft_collection_name - maximum up to 50 characters.

3. Payment

- Payment mode - Metamask wallet, payments are done using Ethereum blockchain.

3.5.3 Data Dictionary

Table 3.7 User Table

Sl. No.	Attribute name	Min	Max	Default
1.	User_ID			auto-generated

Table 3.8 User's NFT details Table

Sl. No.	Attribute name	Min	Max	Default
1.	Nft_ID			auto-generated
2.	Collection_name	1	50	Not null
3.	Image_url			Unique

Table 3.9 NFT details Table

Sl. No.	Attribute name	Min	Max	Default
1.	Image_url			Unique
2.	Collection_name	1	50	Not null

Table 3.10 Rankings Table

Sl. No.	Attribute name	Min	Max	Default
1.	Items	1	50	Not null

Table 3.11 Assets Table

Sl. No.	Attribute name	Min	Max	Default
1.	Image_url			Unique
2.	Collection_name	1	50	Not null
3.	Contract_date			Not null

Table 3.12 Sales Table

Sl. No.	Attribute name	Min	Max	Default
1.	Image_url			Unique
2.	Collection_name	1	50	Not null
3.	Contract_date			Not null
4.	Contract_address		150	Unique

3.6 INTERFACE AND PROCEDURAL DESIGN

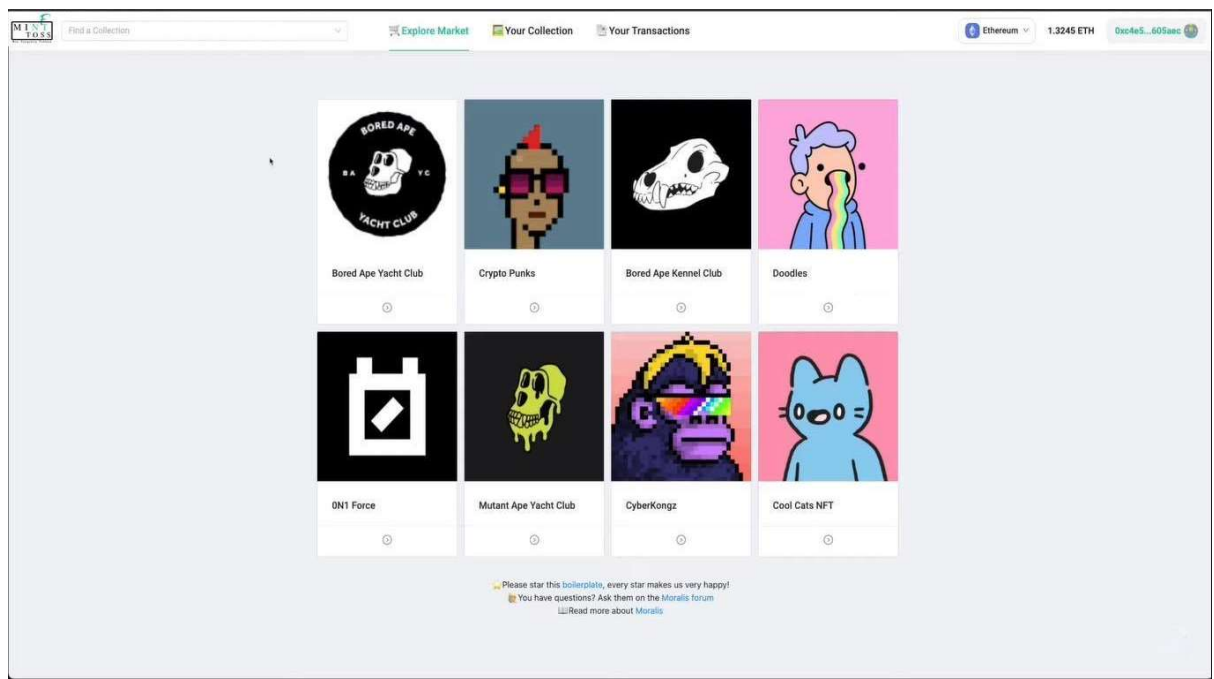


Fig 3.7 Home page

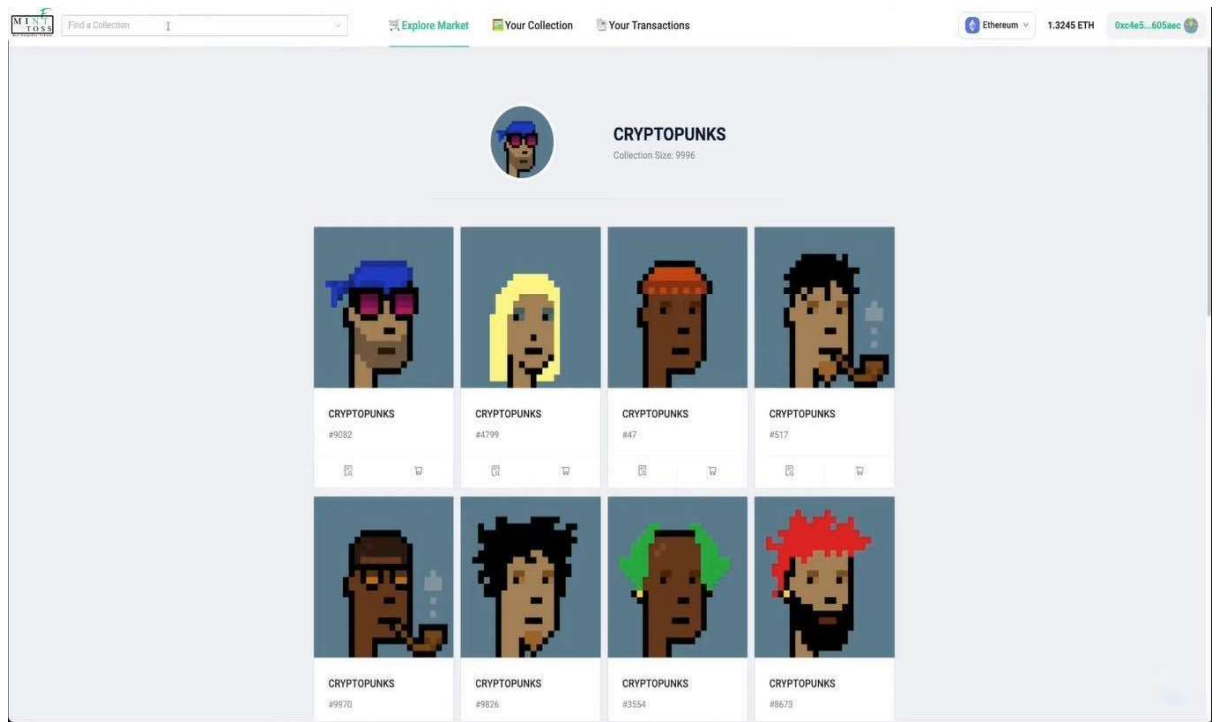


Fig 3.8 Collections page

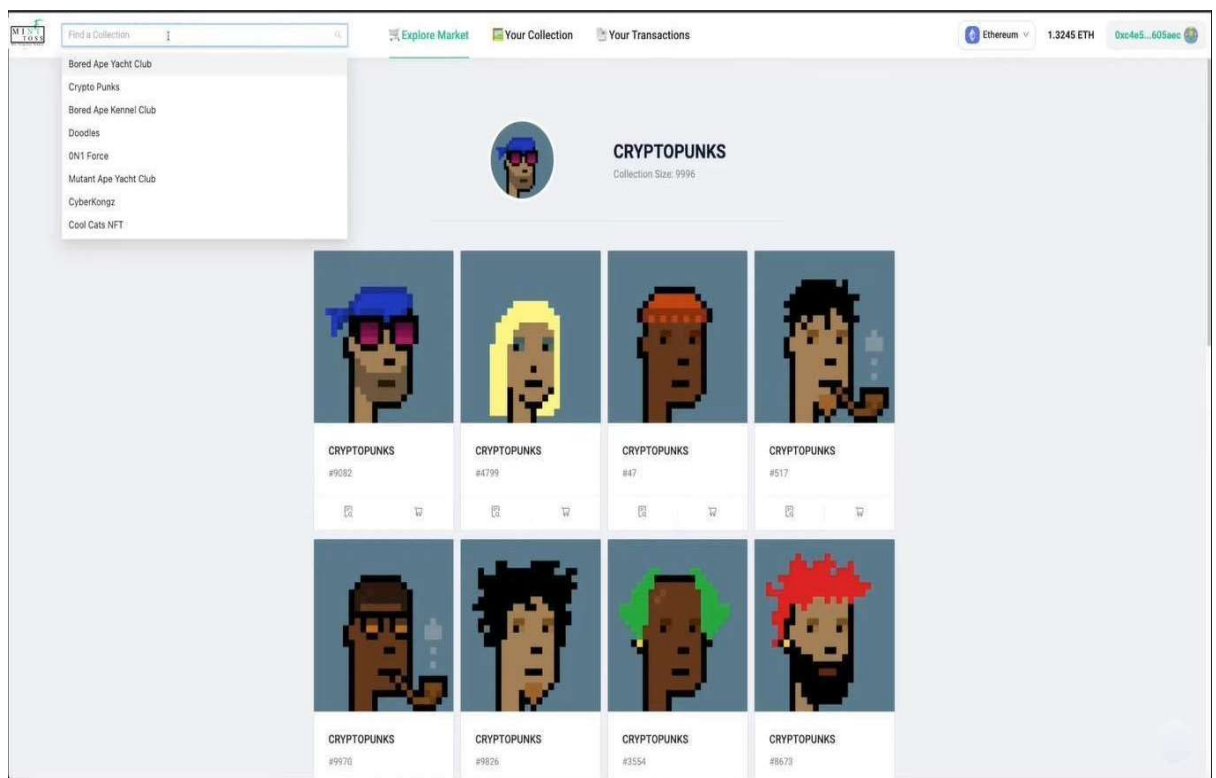
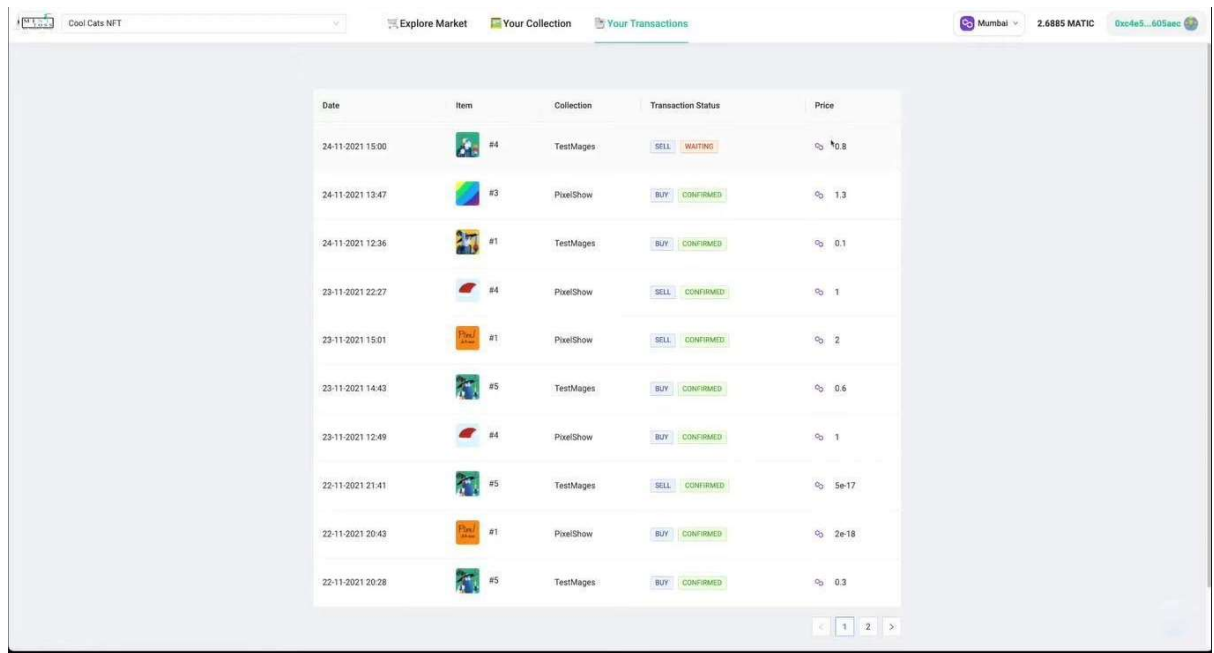


Fig 3.9 Search module



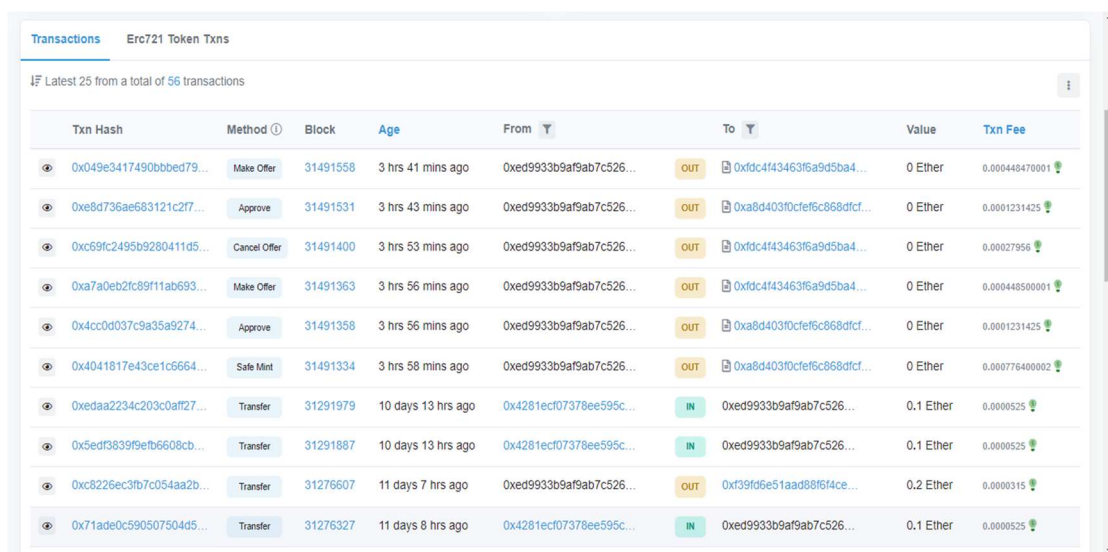
Date	Item	Collection	Transaction Status	Price
24-11-2021 15:00	#4	TestMages	SELL WAITING	0.8
24-11-2021 13:47	#3	PixelShow	BUY CONFIRMED	1.3
24-11-2021 12:36	#1	TestMages	BUY CONFIRMED	0.1
23-11-2021 22:27	#4	PixelShow	SELL CONFIRMED	1
23-11-2021 15:01	#1	PixelShow	SELL CONFIRMED	2
23-11-2021 14:43	#5	TestMages	BUY CONFIRMED	0.6
23-11-2021 12:49	#4	PixelShow	BUY CONFIRMED	1
22-11-2021 21:41	#5	TestMages	SELL CONFIRMED	5e17
22-11-2021 20:43	#1	PixelShow	BUY CONFIRMED	2e18
22-11-2021 20:28	#5	TestMages	BUY CONFIRMED	0.3

Fig 3.10 Transactions page

3.7 REPORTS DESIGN

Design reports are frequently written by engineers to document the process and outcomes of a design task. They communicate to your reader how well you've understood the problem, how you've evolved the design throughout your study, and what the next steps are.

The transaction reports are displayed through the help of Metamask and Ether Scan. It displays the transaction Id, method, time, From wallet address, To wallet address, value (price) and transaction fee.



Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x049e3417490bbbed79...	Make Offer	31491558	3 hrs 41 mins ago	0xed9933b9af9ab7c526...	OUT 0xtdc4f43463f6a9d5ba4...	0 Ether	0.000448470001
0xe8d736ae683121c2f7...	Approve	31491531	3 hrs 43 mins ago	0xed9933b9af9ab7c526...	OUT 0xa8d403f0cfe6c868dfc...	0 Ether	0.0001231425
0xc69fc2495b9280411d5...	Cancel Offer	31491400	3 hrs 53 mins ago	0xed9933b9af9ab7c526...	OUT 0xtdc4f43463f6a9d5ba4...	0 Ether	0.00027956
0xa7a0eb2fc89f11ab693...	Make Offer	31491363	3 hrs 56 mins ago	0xed9933b9af9ab7c526...	OUT 0xtdc4f43463f6a9d5ba4...	0 Ether	0.000448500001
0x4cc0d037c9a35a9274...	Approve	31491358	3 hrs 56 mins ago	0xed9933b9af9ab7c526...	OUT 0xa8d403f0cfe6c868dfc...	0 Ether	0.0001231425
0x4041817e43ce1c6664...	Safe Mint	31491334	3 hrs 58 mins ago	0xed9933b9af9ab7c526...	OUT 0xa8d403f0cfe6c868dfc...	0 Ether	0.000776400002
0xedaa2234c203c0aff27...	Transfer	31291979	10 days 13 hrs ago	0x4281ecf07378ee595c...	IN 0xed9933b9af9ab7c526...	0.1 Ether	0.0000525
0x5edf3839f9eb6608cb...	Transfer	31291887	10 days 13 hrs ago	0x4281ecf07378ee595c...	IN 0xed9933b9af9ab7c526...	0.1 Ether	0.0000525
0xc8226ec3b7c054aa2b...	Transfer	31276607	11 days 7 hrs ago	0xed9933b9af9ab7c526...	OUT 0xf39f06e51aad88f64ce...	0.2 Ether	0.0000315
0x71ade0c590507504d5...	Transfer	31276327	11 days 8 hrs ago	0x4281ecf07378ee595c...	IN 0xed9933b9af9ab7c526...	0.1 Ether	0.0000525

Fig 3.11 Transactions report

4. IMPLEMENTATION

This chapter consists of the coding standards used, the coding details, the code as well as the output of the desired implementation.

4.1 CODING STANDARDS

A coding standard gives a uniform appearance to the codes written by different engineers. It improves readability, and maintainability of the code and it reduces complexity also. It helps in code reuse and helps to detect errors easily. It promotes sound programming practices and increases efficiency of the programmers.

The following coding standards are followed:

- **Indenting** - Used an indent of 4 spaces and didn't use any tab because different computers use different setting for tab.
- **Variable Names** - Used all lowercase letters and used '_' as the word separator.
- **Control Structures** - These include if, for, while, switch, etc. Control statements have one space between the control keywords, to distinguish them from function calls.
- **Function Calls** - Functions have been called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon.
- **Docstrings** - There are both single and multi-line docstrings that can be used in Python. However, the single line comment fits in one-line, triple quotes are used in both cases. These are used to define a particular program or define a particular function.
- **Comments** - There are also various types and conditions that are followed that can be of great help from programs and user's point of view. Comments should form complete sentences. If a comment is a full sentence, its first word should be capitalized, unless it is an identifier that begins with a lowercase letter. In short comments, the period at the end can be omitted. In block comments, there are more than one paragraphs and each sentence must end with a period. Block comments and inline comments can be written followed by a single '#'.
#
- **Alignment of Declaration Blocks** - Block of declarations are be aligned.

- **One Statement Per Line** - There is only one statement per line unless the statements are very closely related.

4.2 CODING DETAILS

Mint NFT

```
useEffect(() => {
  if (!selectedFile) {
    setPreview(undefined);
    return;
  }
  const objectUrl = URL.createObjectURL(selectedFile);
  setPreview(objectUrl);
  return () => URL.revokeObjectURL(objectUrl);
}, [selectedFile]);

const enteredNameHandler = (event) => {
  setEnteredName(event.target.value);
};

const enteredDescriptionHandler = (event) => {
  setEnteredDescription(event.target.value);
};

const onSelectFile = (e) => {
  if (!e.target.files || e.target.files.length === 0) {
    setSelectedFile(undefined);
    return;
  }
  setSelectedFile(e.target.files[0]);
};

// SUBMISSION => submit create form
const submissionHandler = (event) => {
  event.preventDefault();

  // Validate form fields
  enteredName ? setNameIsValid(true) : setNameIsValid(false);
  enteredDescription ? setDescriptionIsValid(true) : setDescriptionIsValid(false);
  selectedFile ? setFileIsValid(true) : setFileIsValid(false);
  const formIsValid = enteredName && enteredDescription && selectedFile;

  // Upload file to IPFS and push to the blockchain
  const mintNFT = async () => {
    setMintLoading(true);

    // Add file to the IPFS
    const fileAdded = await ipfs.add(selectedFile);
    if (!fileAdded) {
      console.error('Something went wrong when uploading the file');
    }
  }
}
```

```
    return;
  }

  const metadata = {
    title: 'Asset Metadata',
    type: 'object',
    properties: {
      name: {
        type: 'string',
        description: enteredName,
      },
      description: {
        type: 'string',
        description: enteredDescription,
      },
      image: {
        type: 'string',
        description: fileAdded.path,
      },
      category: {
        type: 'string',
        description: enteredCategory,
      },
      dateCreated: {
        type: 'string',
        description: today,
      },
    },
  };

  const metadataAdded = await ipfs.add(JSON.stringify(metadata));
  if (!metadataAdded) {
    console.error('Something went wrong when uploading the file');
    return;
  }

  collectionCtx.contract.methods
    .safeMint(metadataAdded.path)
    .send({ from: web3Ctx.account })
    .on('transactionHash', (hash) => {
      collectionCtx.setNftIsLoading(true);
      setMintLoading(true);
    })
    .on('error', (e) => {
      addToast('Something went wrong when pushing to the blockchain', {
        appearance: 'error',
      });
      collectionCtx.setNftIsLoading(false);
      setMintLoading(false);
    })
```

```

        .on('receipt', () => {
            setMintSuccess(true);
            setMintLoading(false);
            setTimeout(() => {
                history.push('/explore');
            }, 2500);
        });
    };

    formIsValid && mintNFT();
};

// Inject validation classes to input fields
const nameClass = nameIsValid ? 'form-control' : 'form-control is-invalid';
const descriptionClass = descriptionIsValid ? 'form-control' : 'form-control is-
invalid';
const fileClass = fileIsValid ? 'form-control' : 'form-control is-invalid';

if (mintSuccess)
    return (
        <SuccessMessage
            heading="Great! You've successfully minted your NFT"
            subheading="We're redirecting to homepage"
        />
    );

```

Listing NFT

```

function NftItem({ img, title, owner, price, category, dateCreated, id, index, nftKey })
{
    const web3Ctx = useContext(Web3Context);
    const collectionCtx = useContext(CollectionContext);
    const marketplaceCtx = useContext(MarketplaceContext);
    const { addToast } = useToasts();

    const priceRefs = useRef([]);
    if (priceRefs.current.length !== collectionCtx.collection.length) {
        priceRefs.current = Array(collectionCtx.collection.length)
            .fill()
            .map((_, i) => priceRefs.current[i] || createRef());
    }

    const makeOfferHandler = (event, id, nftKey) => {
        event.preventDefault();

        const enteredPrice = web3.utils.toWei(priceRefs.current[nftKey].current.value,
'ether');

        collectionCtx.contract.methods
            .approve(marketplaceCtx.contract.options.address, id)

```

```

        .send({ from: web3Ctx.account })
        .on('transactionHash', (hash) => {
            marketplaceCtx.setMktIsLoading(true);
        })
        .on('receipt', (receipt) => {
            marketplaceCtx.contract.methods
                .makeOffer(id, enteredPrice)
                .send({ from: web3Ctx.account })
                .on('error', (error) => {
                    addToast('Something went wrong when pushing to the blockchain', {
                        appearance: 'error',
                    });
                    marketplaceCtx.setMktIsLoading(false);
                });
        });
    });
};

const buyHandler = (event) => {
    const buyIndex = parseInt(event.target.value);
    marketplaceCtx.contract.methods
        .fillOffer(marketplaceCtx.offers[buyIndex].offerId)
        .send({ from: web3Ctx.account, value: marketplaceCtx.offers[buyIndex].price
    })

    .on('transactionHash', (hash) => {
        marketplaceCtx.setMktIsLoading(true);
    })
    .on('error', (error) => {
        addToast('Something went wrong when pushing to the blockchain', {
            appearance: 'error',
        });
        marketplaceCtx.setMktIsLoading(false);
    });
};

const cancelHandler = (event) => {
    const cancelIndex = parseInt(event.target.value);
    marketplaceCtx.contract.methods
        .cancelOffer(marketplaceCtx.offers[cancelIndex].offerId)
        .send({ from: web3Ctx.account })
        .on('transactionHash', (hash) => {
            marketplaceCtx.setMktIsLoading(true);
        })
        .on('error', (error) => {
            addToast('Something went wrong when pushing to the blockchain', {
                appearance: 'error',
            });
            marketplaceCtx.setMktIsLoading(false);
        });
};

```

NFT Category

```

function Category({ category }) {
  return (
    <p className='text-sm text-muted fw-normal mb-2 d-flex align-items-center'>
      <span className='icon bg-primary text-white me-2'>
        <i className='las la-icons fa-fw'></i>
      </span>
      <span>category: </span>
      {category ? (
        <Link className='text-reset' to={`/${categories}/${category}`} >
          <span className='text-primary ms-2'>{formatCategory(category)}</span>
        </Link>
      ) : (
        <span className='text-white ms-2'>No Cateogry</span>
      )}
    </p>
  );
}

return (
  <div>
    {marketplaceCtx.mktIsLoading ? <FullScreenLoader heading='loading' /> :
    null}
    <PageBanner heading={` ${formatCategory(category)} NFTs` } />
    <section className='py-5'>
      <div className='container py-5'>
        {collectionCtx.collection.length !== 0 && collectionCtx.totalSupply !==
        0 ? (
          <div className='row mixitUpContainer gy-4 mb-5 align-items-
          stretch'>
            {currentItems
              .filter((el) => el.category === category)
              .map((NFT, key) => {
                const index = marketplaceCtx.offers
                  ? marketplaceCtx.offers.findIndex((offer) => offer.id ===
                  NFT.id)
                  : -1;
                const owner = index === -1 ? NFT.owner :
                marketplaceCtx.offers[index].user;
                const price =
                  index !== -1
                  ?
                formatPrice(marketplaceCtx.offers[index].price).toFixed(2)
                  : null;

                return (
                  <div className={`col-xl-3 col-lg-4 col-md-6 mix
                  ${NFT.category}`} key={key}>

```



```

        <NftItem {...NFT} index={index} owner={owner}
price={price} nftKey={key} />
      </div>
    );
  }}}
</div>
): (
  <
    <h6 className='fw-normal text-muted text-center mb-0'>
      Fetching data from the blockchain please wait...
    </h6>
    <Loader />
  </>
)
)

{ /* <Pagination
  itemsPerPage={itemsPerPage}
  totalItems={collectionCtx.collection.length}
  paginate={paginate}
  currentPage={currentPage}
/> */}

{currentItems.filter((el) => el.category === category).length === 0 &&
  collectionCtx.collection.length !== 0 &&
  collectionCtx.totalSupply !== 0 && (
    <div className='row text-center'>
      <div className='col-lg-6 mx-auto'>
        <i className='las la-exclamation mb-2' style={{ fontSize:
'3rem' }}></i>
        <h3 className='h3'>No NFTs listed under this category.</h3>
        <p className='text-muted mb-3'>Return Home and pick
another category...</p>
        <Link className='btn btn-gradient-primary' to='/>
          Return Home
        </Link>
      </div>
    </div>
  )}
</div>
</section>
</>
);
}

```

My Assets

```

function MyAssets() {
  const collectionCtx = useContext(CollectionContext);
  const marketplaceCtx = useContext(MarketplaceContext);

```

```

const web3Ctx = useContext(Web3Context);
const [currentAddress, setCurrentAddress] = useState();

useEffect(() => {
  document.title = 'My assets';
  setCurrentAddress(web3Ctx.account);
}, [web3Ctx.account]);

// RETURN ITEMS TEMPLATE
return (
  <
    {marketplaceCtx.mktIsLoading ? <FullScreenLoader heading='loading' /> :
  null}
    <PageBanner heading={'My Assets'} />

    {/* NFT ITEMS */}
    <section className='py-5'>
      <div className='container'>
        <header className='mb-5'>
          <div className='row'>
            <div className='col-lg-6'>
              <h2>Your available NFTs</h2>
              <p className='text-muted text-sm mb-0'>
                The NFTs you own are listed here!
              </p>
            </div>
          </div>
        </header>

        <div className='row gy-4'>
          {collectionCtx.collection
            .filter((item) => item.owner === currentAddress)
            .map((NFT, key) => {
              const index = marketplaceCtx.offers
                ? marketplaceCtx.offers.findIndex((offer) => offer.id ===
NFT.id)
                : -1;
              const owner = index === -1 ? NFT.owner :
marketplaceCtx.offers[index].user;
              const price =
                index !== -1 ?
formatPrice(marketplaceCtx.offers[index].price).toFixed(2) : null;

              return (
                <div className={`col-xl-3 col-lg-4 col-md-6 mix`}
key={key}>
                  <NftItem {...NFT} index={index} owner={owner}
price={price} nftKey={key} />
                </div>
              );
            })}
        </div>
      </div>
    </section>
  </

```

```

    }}}
    {collectionCtx.collection.filter((item) => item.owner ===
currentAddress).length === 0 ? (
      <div className='col-9'>
        <NoDataAlert
          heading="You don't have any assets that lays under your
ownership, if you put assets for
          sale you might see them at the section below."
          subheading='Please note that when you put item for sale the
ownership goes to
          marketplace and the marketplace will sell it for you.'
        />
      </div>
    ) : null}
  </div>
</div>
</section>

```

Ranking Authors

```

const renderSellers = currentAuthors
  .sort((a, b) => (a.value < b.value ? 1 : -1))
  .map((seller, index) => {
    return (
      <div className='col-xl-3 col-lg-4 col-md-6' key={index}>
        <div className='card bd-3 card-hover-minimal position-relative'>
          <div className='card-body'>
            <a
              className='d-flex align-items-center text-reset text-decoration-
none stretched-link'
              href={configEtherScanUrl(web3Ctx.networkId, seller.address)}
              rel='noreferrer noopener'
              target='_blank'
            >

            <div className='position-relative'>
              <div className='ms-3' style={{ width: '50px', height: '50px'

              <Jazzicon address={seller.address} />
            </div>
            <div className='author-img-badge bg-primary text-white'>
              <i className='las la-check-double la-xs'></i>
            </div>
          </div>
          <div className='ms-3'>
            <h3 className='h6 mb-1 text-capitalize'>
              {uniqueNamesGenerator({
                dictionaries: [starWars],
              }).replace('_', ' ')}

```

```

        {web3Ctx.account === seller.address ? (
          <span className='seller-badge ms-2'>You</span>
        ) : null}
      </h3>
      <p className='text-sm text-primary mb-0'>
        {formatPrice(seller.value).toFixed(2)} <span
className='text-muted'>ETH</span>
      </p>
    </div>
  </a>
</div>
</div>
</div>
);
});

```

Contact form

```

function ContactForm({ gridWidth }) {
  const [state, handleSubmit] = useForm('xnqwjpgvp');
  useEffect(() => {
    // Fetch all the forms we want to apply custom Bootstrap validation styles to
    var forms = document.querySelectorAll('.needs-validation');

    // Loop over them and prevent submission
    Array.prototype.slice.call(forms).forEach(function (form) {
      form.addEventListener(
        'submit',
        function (event) {
          if (!form.checkValidity()) {
            event.preventDefault();
            event.stopPropagation();
          }

          form.classList.add('was-validated');
        },
        false
      );
    });
  }, []);
  if (state.succeeded) {
    return (
      <div className={` ${gridWidth} text-center`} >
        <p className='mb-0 fw-bold mt-5 mb-0'>
          <i
            className='las la-grin-beam'
            style={{ fontSize: '10rem', textShadow: '2px 4px rgba(0, 0, 0, 0.4)' }}
          ></i>
        </p>
      </div>
    );
  }
}

```

```
    <h1 className='h2'>Thanks for contacting us.</h1>
    <p className='text-muted'>We'll reply back as soon as possible.</p>
    <Link to='/' className='btn btn-gradient-primary'>
      Return Home
    </Link>
  </div>
);
}
```

4.3 SCREENSHOTS

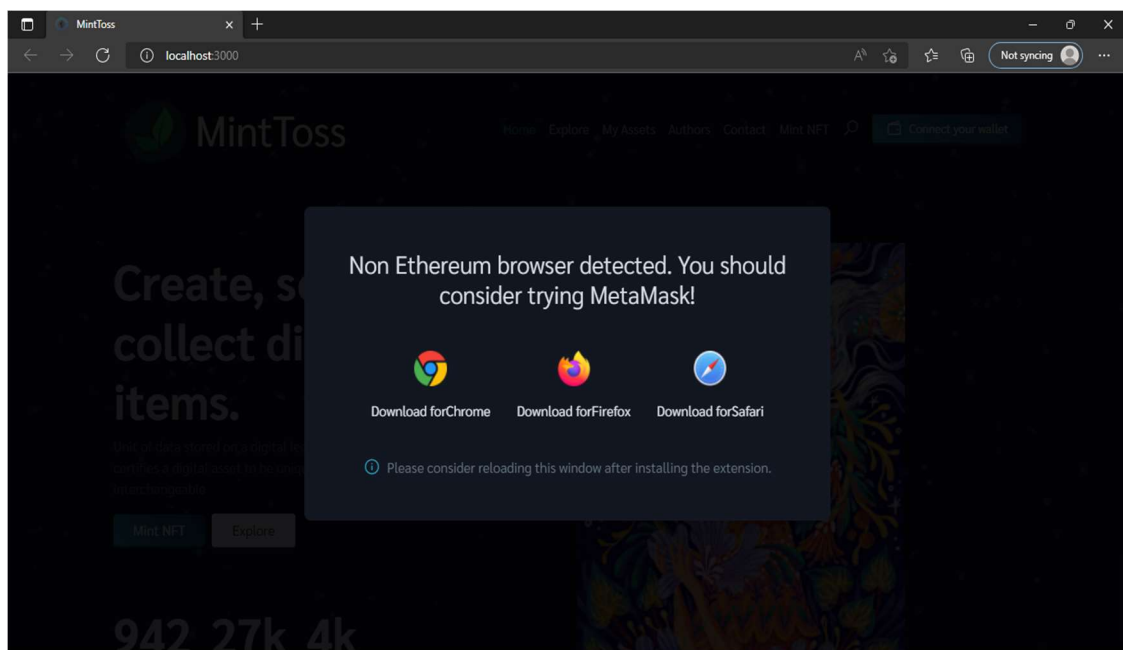


Fig 4.1 Metamask Authentication

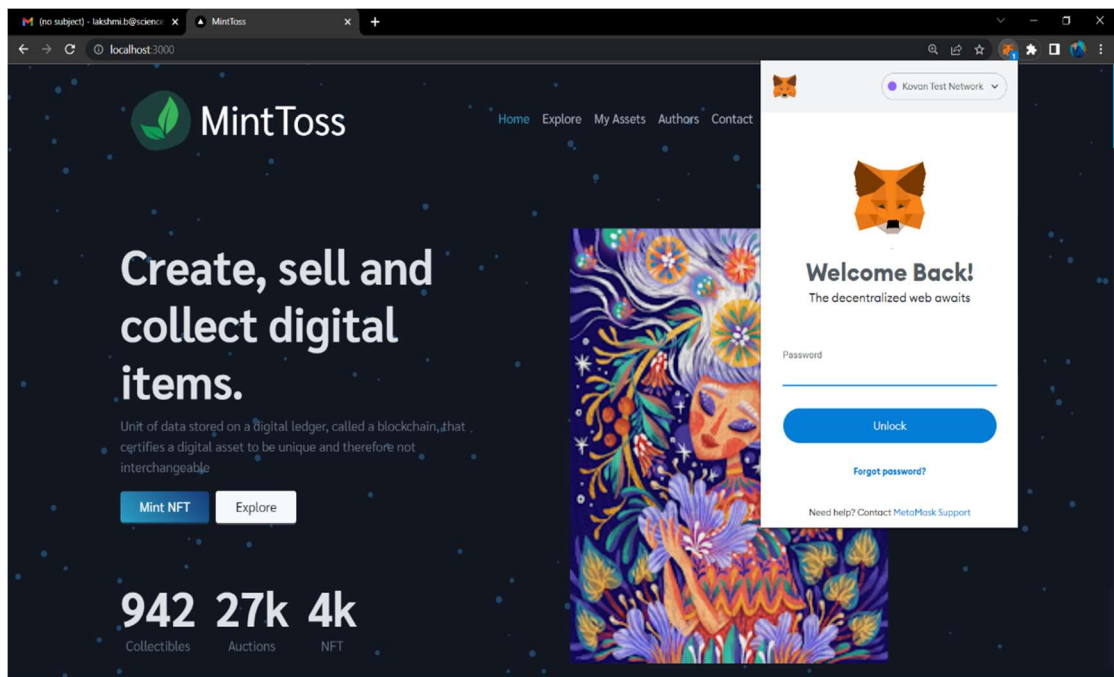


Fig 4.2 Metamask Login

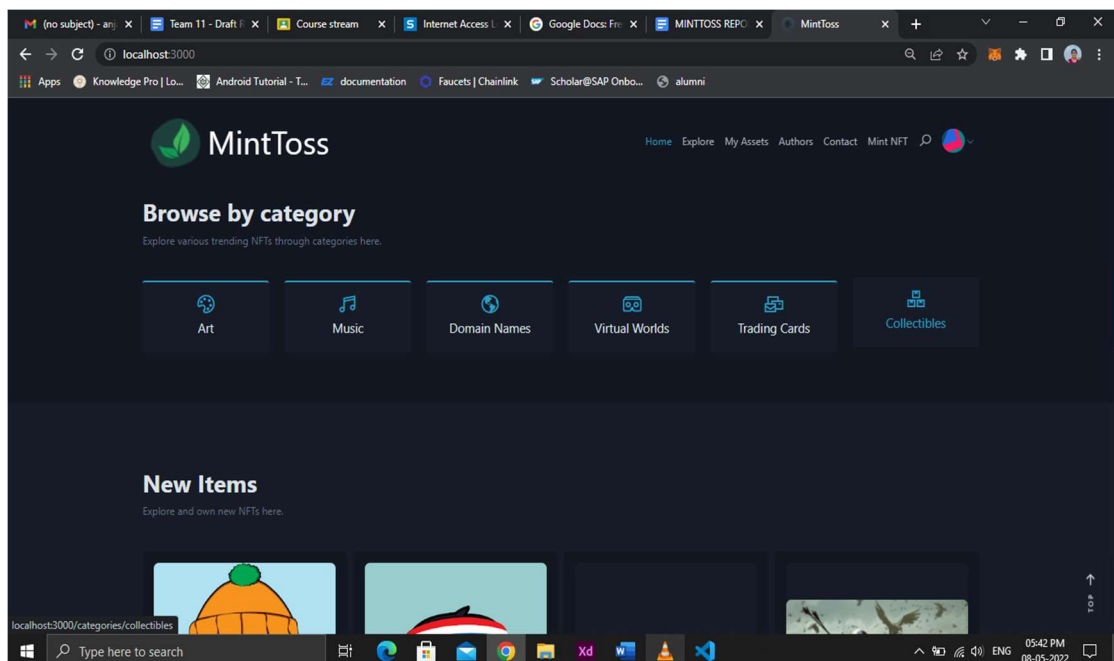


Fig 4.3 Home page – Browse by category

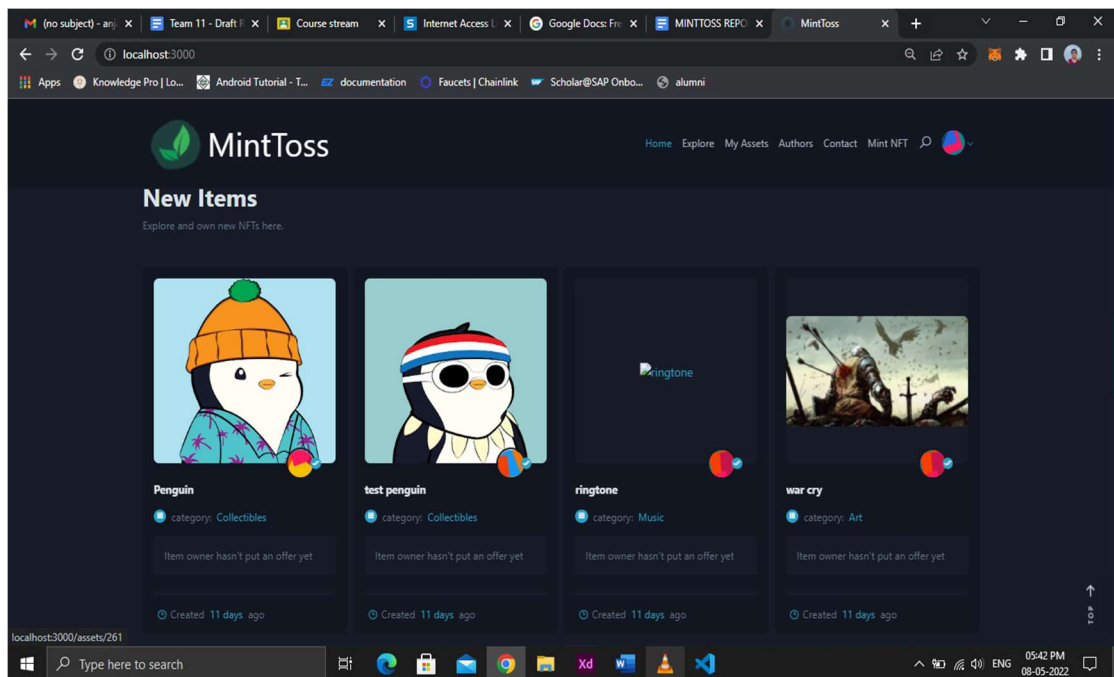


Fig 4.4 Home page - New items

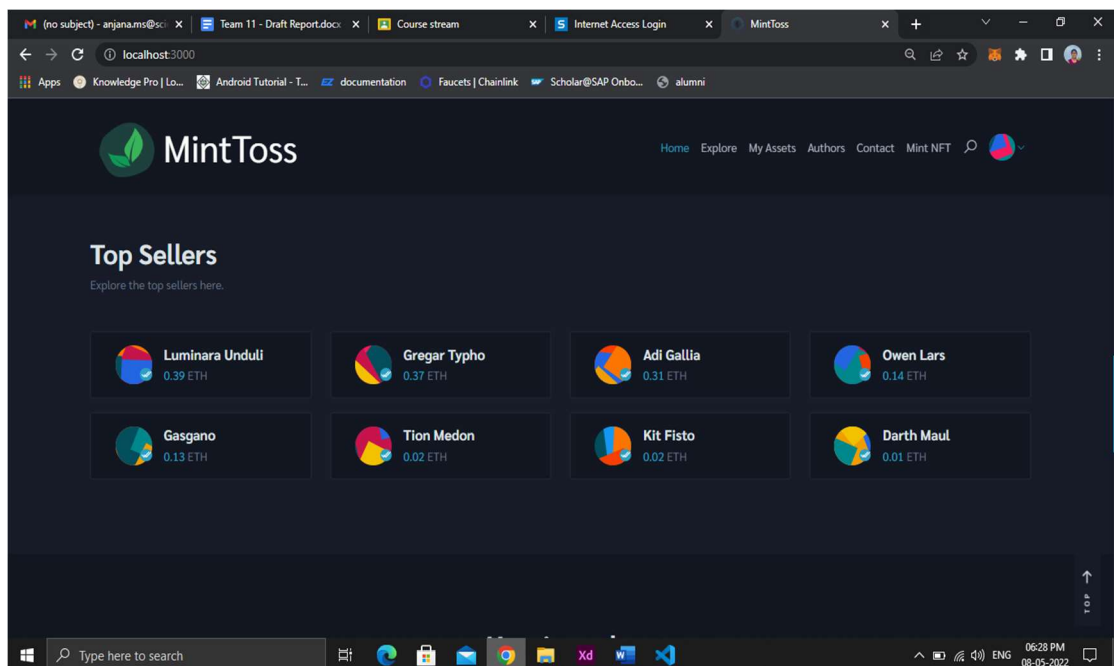


Fig 4.5 Home page –Top sellers

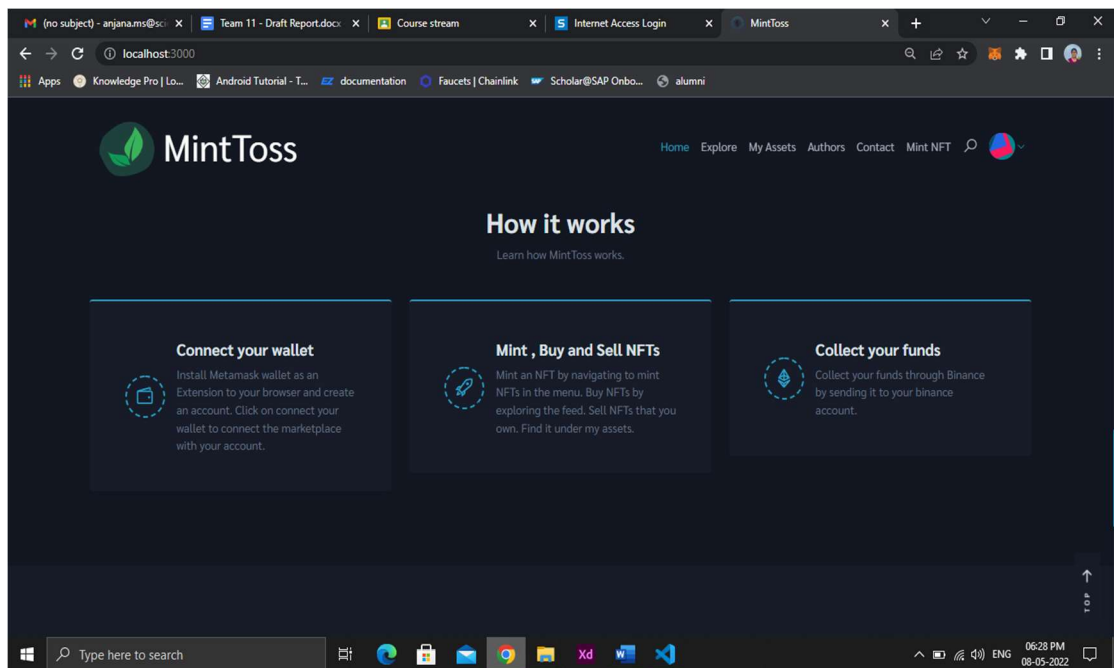


Fig 4.6 Home page –How it works?

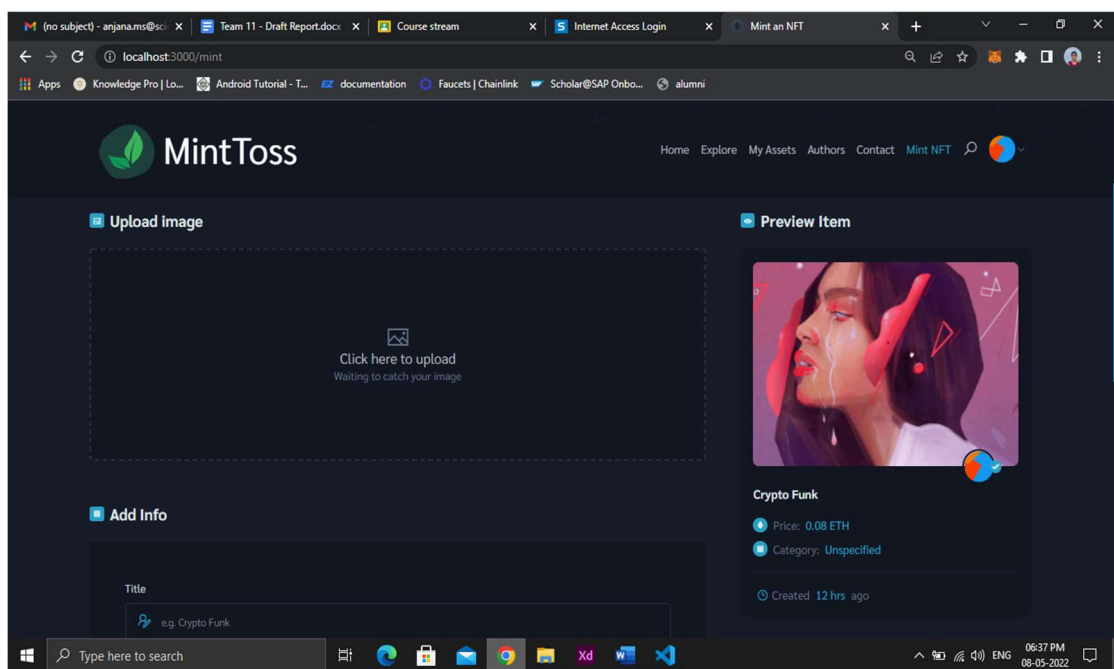


Fig 4.7 Mint NFT Form

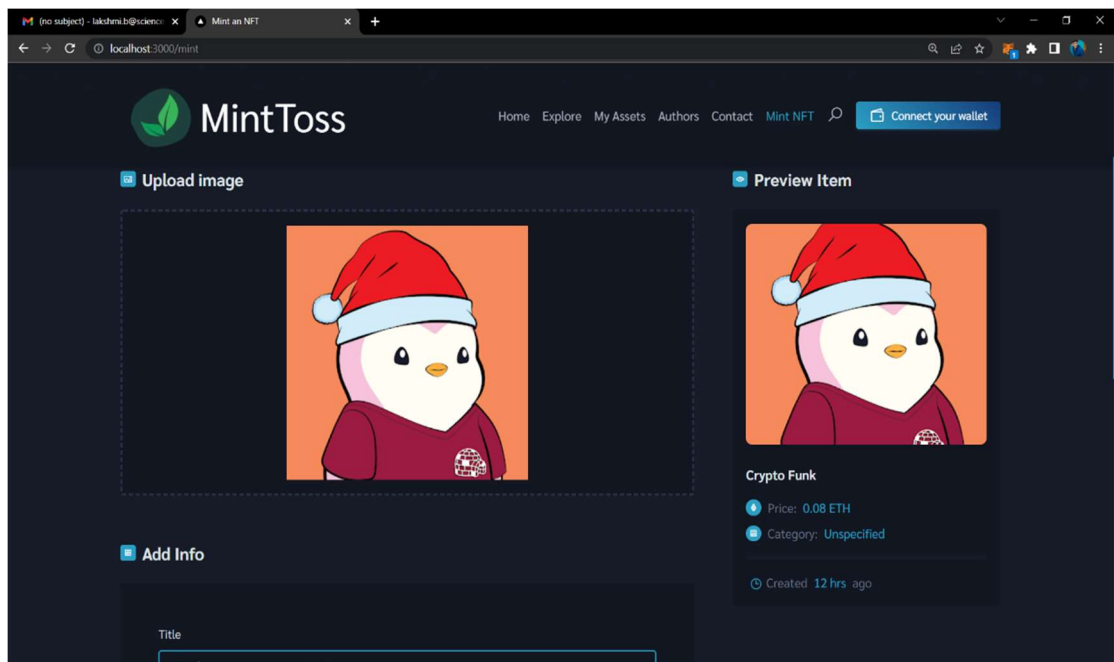


Fig 4.8 Minting NFT with Preview

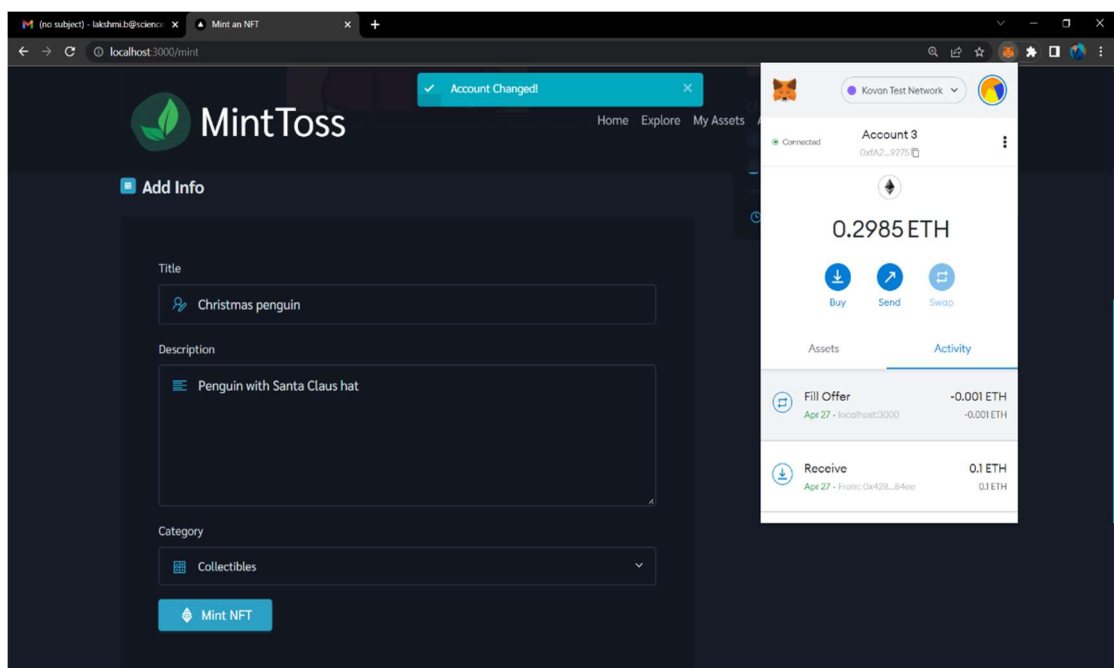


Fig 4.9 NFT Description

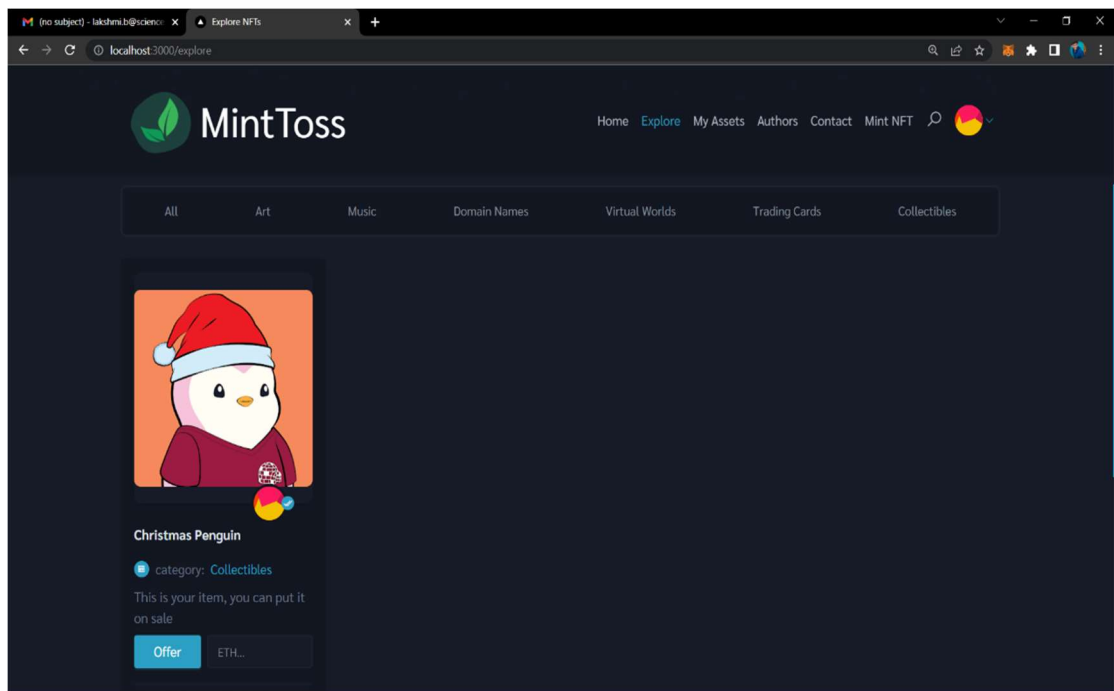


Fig 4.10 Minted NFT – Ready to be listed

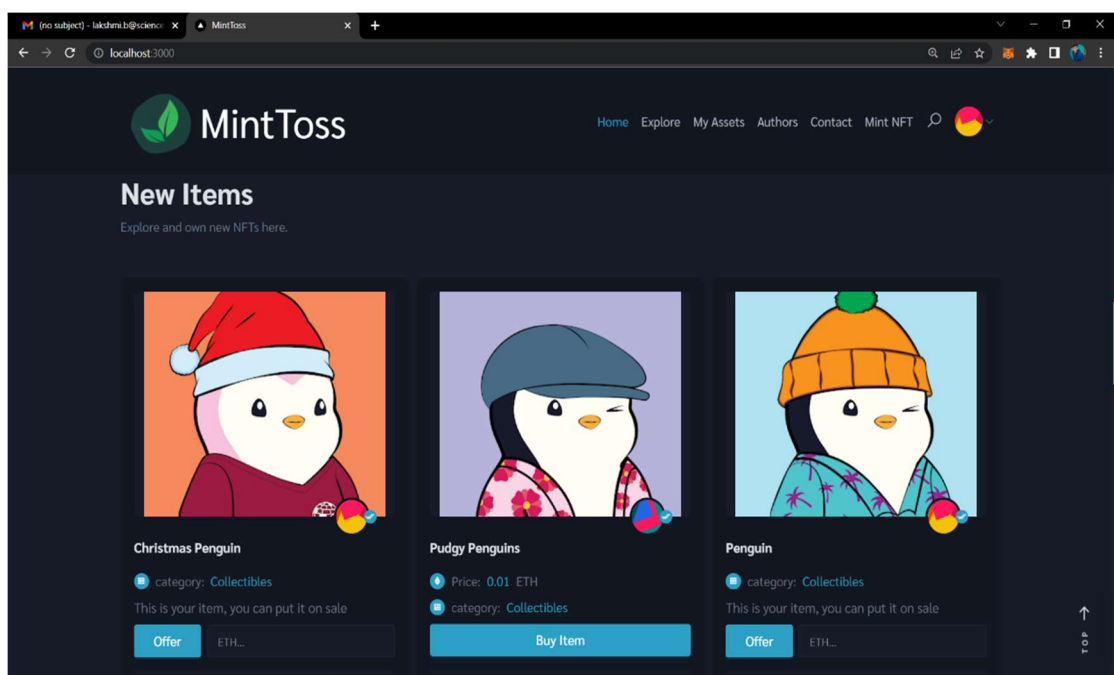


Fig 4.11 NFT listed on new items

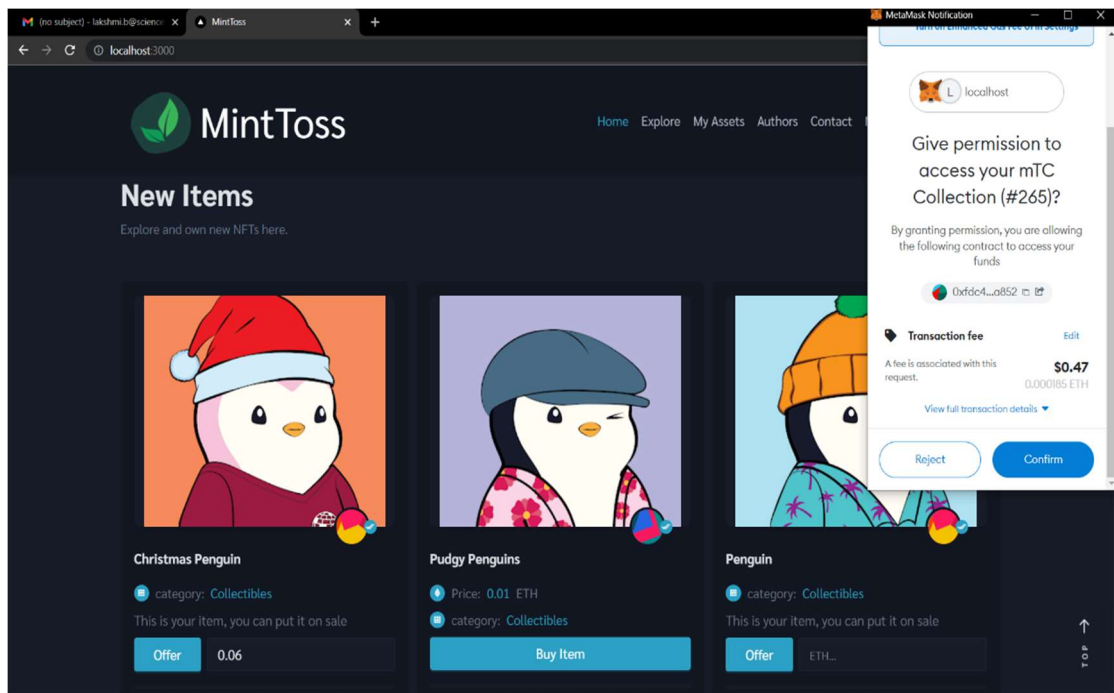


Fig 4.12 Listing NFT out on the marketplace

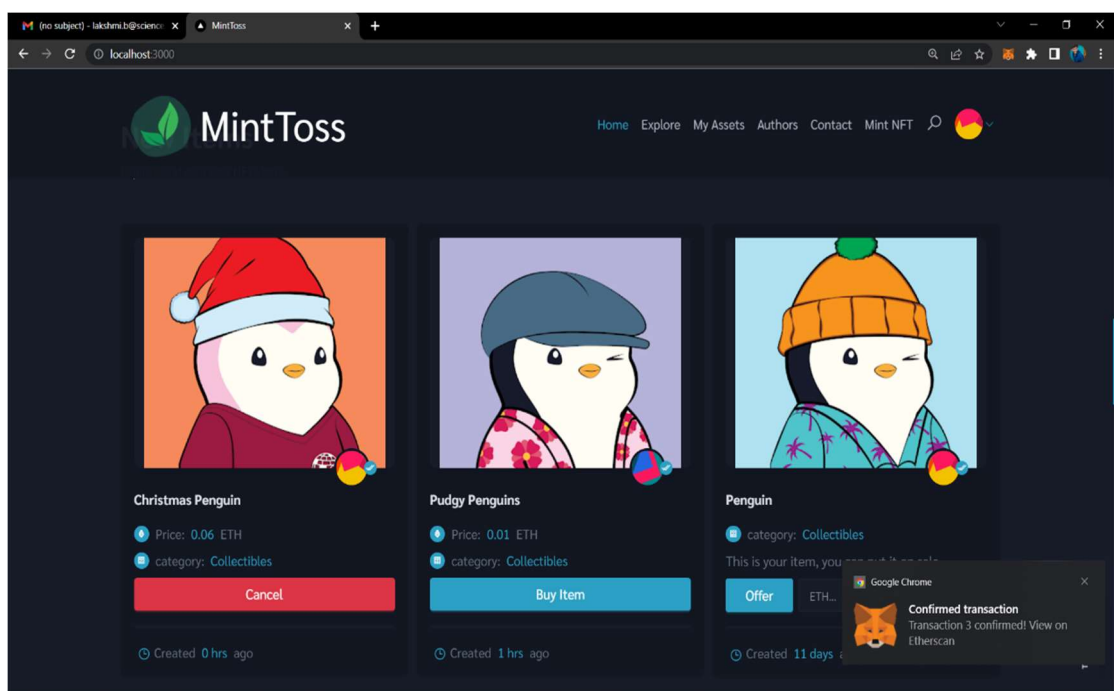


Fig 4.13 Listed NFT

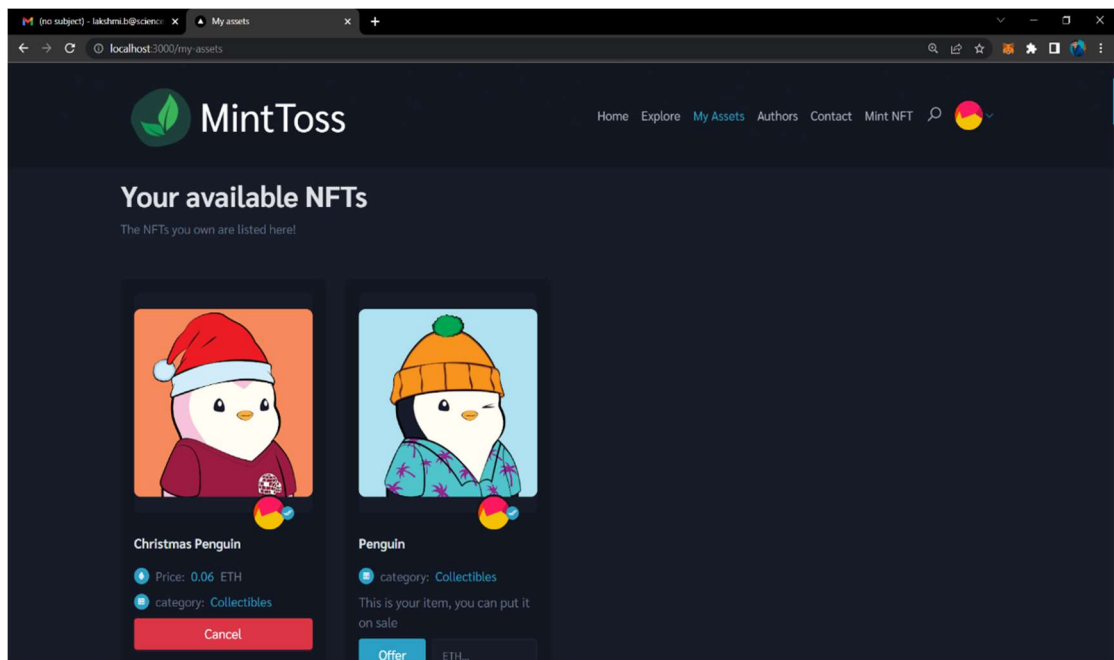


Fig 4.14 My Assets

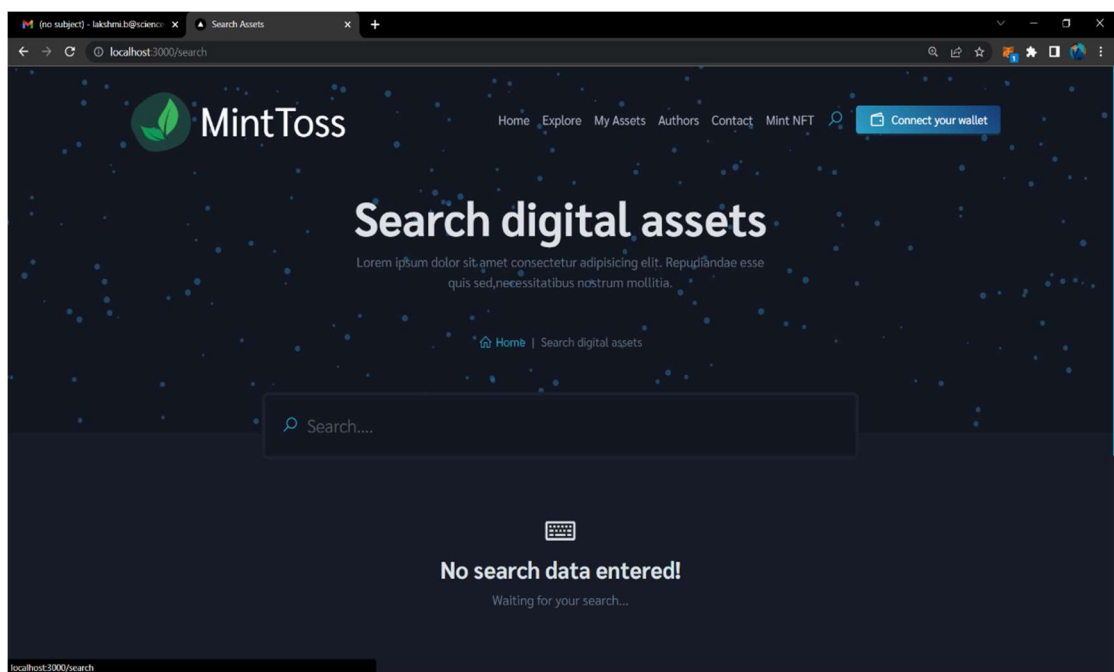


Fig 4.15 Search page

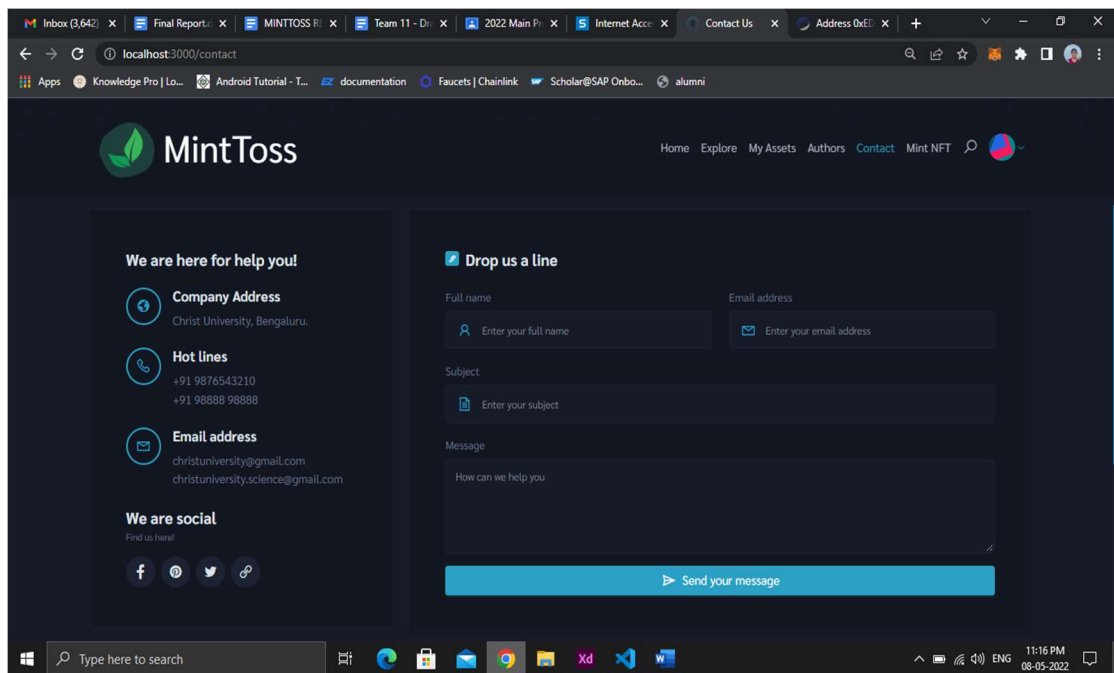


Fig 4.16 Contact Us page

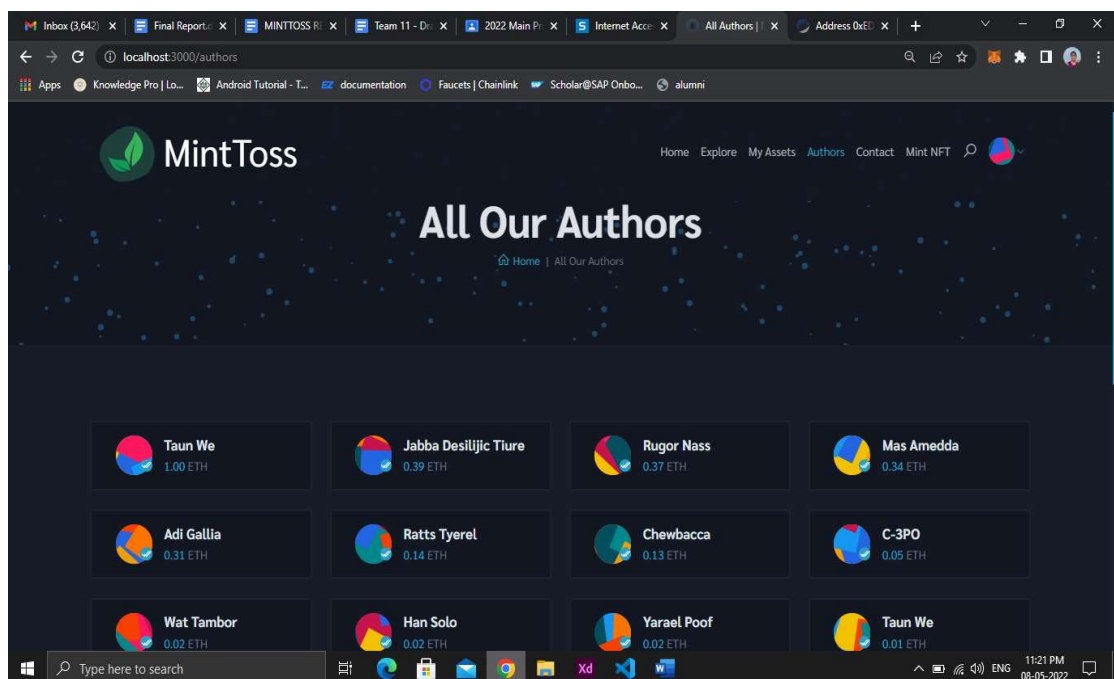


Fig 4.17 Ranking Authors page

5. TESTING

5.1 TEST STRATEGIES

5.1.1 System Testing

System testing is a critical element of quality assurance and represents the ultimate review of analysis, design and coding. Test case design focuses on set of techniques for the creation of test because that meet overall testing objectives. When a system is developed it is hoped that it performs properly. The main purpose of testing an information system is to find the errors and correct them. The scope of system testing should include both manual and computerized operations. System testing is a comprehensive evaluation of the programs, manual procedures, computer operations and controls.

System testing is the process of checking whether the developed system is working according to the objective and requirement. All testing is to be conducted in accordance to the test conditions specified earlier. This will ensure that the test coverage meets the requirements and that testing is done in a systematic manner.

The process of analysis of the software item to detect the differences between existing or required condition and evaluate the features of the software items. The thorough testing of the system before release of the software becomes devoid of bugs and uses minimum space requirements as well as minimum time to perform. The test cases were selected beforehand with expected results recorded for comparison. The selection of the test cases is done vide “White Box Testing” technique to check software requirement fulfilment with intension of finding maximum number of errors with minimum effort and time. Although test cases are a design by considering the cyclomatic complexity, conditional test, still the software code is not in its optional form ,as all other possible alternative parts in the software are not considered .At the integration level, the software will be passing to the third party tests which would further enhance the software optimality and efficiency.

5.1.2 Test Data Implementation

The quality and standardization of the software /application package depends truly on the various predefined testing norms and on the performance of the software over those

norms. There are various standards existing in the software industry the engineered end product

Strives to achieve viz. ISO 9002 SEI CMM Level5 etc. These standards are achieved only when the concerned software fulfils the tests as per the respective norms predefined in them vide the various test cases and parameters using the CASE topologies. Generally, Software is tested both on a stand-alone mode as well after integrating all the modules in the system vide deferent available testing methods/norms. The following Flow Graph methodology was used while testing the software:

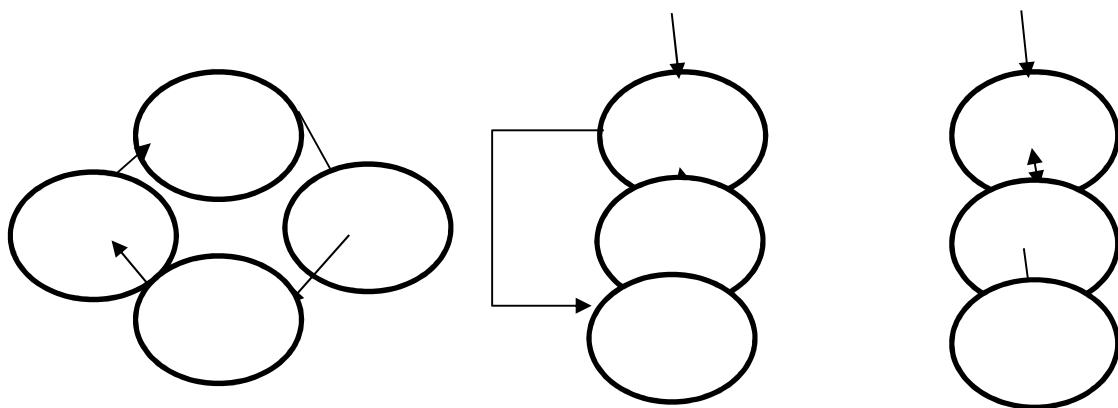


Fig 5.1 Flow graph

Here each circle represents one or more non branching procedural language or source code statements in Flow Graph. While performing Condition Testing Domain Testing methodology was selected. While performing Loop testing simple loops, concatenated loops, nested and unstructured loops were tested thoroughly.

5.1.3 Test Characteristics

1. A good test has a high probability of finding an error.
2. A good test is not redundant.
3. A good test should be “best of breed “.
4. A good test should be neither too simple nor too complex.

5.1.4 Black box Testing

The method of Black Box Testing is used by the software engineer to derive the required results of the test cases:

1. Black Box Testing alludes to test that are conducted at the software interface.

2. A Black Box Test examines some fundamental aspect of a system with little regard for the internal logic structure of the software.
3. A limited number of important logical paths can be selected and exercised.
4. Important data structure can be probed for validity.

Black box testing was performed to find errors in the following categories:

- Incorrect or missing functions
- Graphics errors.
- Errors in data in string format.
- Error in data in integer format.
- File error.
- Memory access error
- Variable error.
- Performance error.

5.1.5 White box Testing

White Box Testing is sometimes called as Glass Box Testing. Using White Box Testing methods, the software engineer can derive the following test cases:

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds.
4. Exercise internal data structures to ensure the validity.

In white Box Testing efforts were made to handle the following:

- Number of input parameters equal to number of arguments.
- Parameters and arguments attributes match.
- Number of arguments transmitted is called modules equal to attributes of parameter.
- Unit system of argument transmitted is called modules equal unit system of parameter.
- Number of attributes and order of arguments to build in functions correct.
- Any references to parameters not associated to build in functions correct.
- Input only arguments altered.
- Global variable definition consistent across module.

- Files attributes correct.
- Format specification matches I/O specification.
- Files opened before use.
- File closed while working.
- I/O errors handled.
- Any textual errors in output information.

Unit Testing

The unit testing is performed to test the validity of the individual units. This is done in the coding phase with the interactive testing. Thus, it itself constitutes a majority of functionality test for each logical unit.

Integrity Testing

When all the development of all the units or modules is completed and integrated, the integrity test phase is started. In this phase, the interface between the modules are tested. This phase basically verifies whether inter module exchange of information and events are as per required system behavior.

Validation Testing

Tests were performed to find conformity with the requirements. Plans and procedures were designed to ensure that all the functional requirements are satisfied. The software was alpha-tested. There are two goals in preparing test plans. Firstly, a properly detailed test plan demonstrates that the program specifications are understood completely. Secondly, the test plan is used during program testing to prove the correctness of the program.

5.2 TEST CASES

Table 5.1 Test Cases

Sl. No.	Module Name	Test Case No	Test Case Description	Expected Result
1	Wallet Extension module	TC1	Verify if the user has a Metamask Extension.	The user must have installed Metamask as an Extension to their browser.
2	Connect wallet module	TC2	While connecting wallet, verify if the user has an account in the Metamask wallet.	The user must have an account in the wallet.
3	Connect wallet module	TC3	Verify if the user has connected the wallet to the marketplace.	The user must connect their wallet to the marketplace in order to avail the services of the site.
4	Balance module	TC4	Verify if the user has enough Ethereum as balance before minting, buying or listing NFTs.	The user must have enough Ethereum in their wallet as balance.
5	Listing module	TC5	Verify if the user has entered the offer price before listing the NFT on the marketplace.	The owner of the NFT must enter an offer price before listing it on the marketplace.
6	Contact Us module	TC6	Verify if the name is entered in the full name field.	The user must enter his or her full name in the given field.
7	Contact Us module	TC7	Verify if the user has entered a valid Email id.	The user must enter a valid Email Id.
8	Contact Us module	TC8	Verify if the user has entered a message in the message field.	The user must enter a message in order to submit the form.

5.3 TEST REPORTS

Table 5.2 Test Reports

Sl. No.	Test Case No.	Test Status	Test Result
1.	TC1	Successful	Verifies if the user has a Metamask Extension if not it asks the user to install Metamask providing the link.
2.	TC2	Successful	Verifies if the user has a Metamask account or not.
3.	TC3	Successful	Verifies if the user has connected his or her account to the marketplace.
4.	TC4	Successful	Checks if the user has enough Ethereum in his or her account if not message will be displayed saying No enough Ethereum in your account.
5.	TC5	Successful	Checks if the user has entered the offer price.
6.	TC6	Successful	Checks if the user has entered a name in the field.
7.	TC7	Successful	Checks if the user has entered a valid Email id.
8.	TC8	Successful	Checks if the user has entered a message in the message box while submitting.

6. CONCLUSION

The major implementation and design issues along with advantages and disadvantages of the project is properly mentioned. The future scope of the project is also mentioned in this chapter.

6.1 DESIGN AND IMPLEMENTATION ISSUES

Poor design and/or implementation can cause failure or rejection of a software system.

6.1.1 Design Issues

During the design phase, various issues were faced in the designs. Integrating solidity was a little difficult since it was all new. To create smart contracts initially Thirdweb was used which availed the services of creating smart contracts for the NFTs. Since, Thirdweb was helping us do the major part of the system, Openzeppelin was used to create smart contracts instead of depending upon a third-party service like Thirdweb. The alignment of the banner and the full screen loader was a little difficult. Alignment issues of the pictures and banners were faced.

6.1.2 Implementation Issues

Since Thirdweb was used initially to create smart contracts for the NFTs, the minting and listing of NFTs were done in the third-party site where the user has to upload the NFTs in that link using their same Metamask wallet account that is connected in the marketplace. Later, Openzeppelin was used to overcome this issue. So that the user can mint and list NFTs in the marketplace itself without having to visit another site for this.

6.2 ADVANTAGES AND LIMITATIONS

6.2.1 Advantages

- The user can mint, buy and sell NFTs using this marketplace.
- User-friendly interface.
- Ranking of authors of the NFTs are displayed.
- Various sub-headings like Marketplace's choice, New Items, Browse by Category are shown for the users to easily categorize listed NFTs

- NFTs are categorized into various domains like art, music, trading cards, collectibles and so on.

6.2.2 Limitations

- The user must have Metamask Extension in their browser to view the marketplace.
- The user must have a Metamask account and must connect it to the marketplace in order to avail the services.
- Currently, this marketplace only allows Ethereum currency as a mode of payment.

6.3 FUTURE AND SCOPE OF THE PROJECT

In this modern era, artists and creators find it quite hard to find places for listing their artworks in art galleries and exhibitions. This gave them the need to digitize their works. They took the route of a digital network to earn their credits. NFTs provide a great opportunity for artists and creators to monetize their assets. They can directly sell their NFTs to the buyers. But they need a forum to reach those potential buyers.

An NFT marketplace acts as a bridge that gaps the distance between the artists and the commoners. The creators can list their NFTs in the NFT platforms. Interested buyers will approach them to buy the NFT collections. The NFT Marketplace Development is created with the support of blockchain networks, and all the transactions and activities are decentralized. Moreover, it is a great platform to earn more revenue. The revenue for the NFT platform comes through the listing fee, gas fee, bedding fee, and selling fee.

REFERENCES

- [1] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. Eighth Edition. McGraw Hill Education, 2019.
- [2] Dr. Richard Hall Thayer, Dr. Merlin Dorfman, Mr. Leonard L. Tripp, Dr. Friedrich L. Bower. *Software Engineering Essentials: Volume I, Software Management Training*. Fourth Edition, 2012
- [3] Ogal, D., *How To Create An NFT Marketplace Website?*. *ergonized.com*. 20 January 2022. <<https://www.ergonized.com/blog/how-to-create-an-nft-marketplace-website/>> .
- [4] Hardhat.org. 2021. *Ethereum development environment for professionals by Nomic Foundation*. 2 February 2022. <<https://hardhat.org/tutorial/>>.
- [5] Faucets.chain.link. 2022. *Faucets Chainlink*. 14 February 2022. <<https://faucets.chain.link/rinkeby>>.
- [6] Tutorial, S., 2022. *Solidity Basics - Coding in Solidity - Intellipaat*. Intellipaat Blog. 27 March 2022. <<https://intellipaat.com/blog/tutorial/blockchain-tutorial/what-is-solidity/>>
- [7] OpenZeppelin. 2017. *OpenZeppelin Contracts*. 3 April 2022. <<https://openzeppelin.com/contracts/>> .
- [8] ethereum.org. 2021. *Smart contracts* ethereum.org. 7 February 2022. <<https://ethereum.org/en/smart-contracts/>>