**Final Year Project Report**

**“NFT Marketplace”**



Bahria University Islamabad

**Supervisor**

Mr. Ali Irfan

**Submitted by**

Fraz Naveed

01-134181-021

**Department of Computer Science,**

Bahria University, Islamabad.

Table of Contents

[**Certificate** 5](#_Toc102219914)

[**Abstract** 6](#_Toc102219915)

[**Acknowledgments** 7](#_Toc102219916)

[**Table of figures** 8](#_Toc102219917)

[**Table of tables** 8](#_Toc102219918)

[**Acronyms and abbreviations:** 9](#_Toc102219919)

[**Introduction** 10](#_Toc102219920)

[1.1. Introduction: 11](#_Toc102219921)

[1.1.1. Creator Perspective: 11](#_Toc102219922)

[1.1.2. Buyer Perspective: 11](#_Toc102219923)

[1.2. Objective: 11](#_Toc102219924)

[1.3. Problem Description: 12](#_Toc102219925)

[1.4. Methodology: 12](#_Toc102219926)

[1.4.1. Lifecycle 12](#_Toc102219927)

[1.4.1.1. Architecture: 13](#_Toc102219928)

[1.5. Project Scope: 14](#_Toc102219929)

[1.5.1. Justification: 14](#_Toc102219930)

[1.5.2. Deliverables: 15](#_Toc102219931)

[1.5.3. Exclusion: 15](#_Toc102219932)

[1.6. Feasibility Study: 15](#_Toc102219933)

[1.6.1. Risks Involved: 15](#_Toc102219934)

[1.6.2. Resource Requirement: 16](#_Toc102219935)

[1.7. Solution Application Areas: 16](#_Toc102219936)

[1.8. Tools/Technology: 16](#_Toc102219937)

[1.9. The expertise of the Team Members: 17](#_Toc102219938)

[1.10. Milestones: 17](#_Toc102219939)

[**Literature Review** 18](#_Toc102219940)

[2.1. Literature Review: 19](#_Toc102219941)

[2.1.1. Insight on NFTs and the work done: 19](#_Toc102219942)

[2.1.2. Conclusion: 20](#_Toc102219943)

[**Requirement Specification** 21](#_Toc102219944)

[3.1. Document Convention: 22](#_Toc102219945)

[3.2. Intended Audience: 22](#_Toc102219946)

[3.3. Marketplace Activities: 22](#_Toc102219947)

[3.4. User Characteristics and class: 23](#_Toc102219948)

[3.4.1. User Functionalities: 24](#_Toc102219949)

[3.4.2. Creator Functionalities: 24](#_Toc102219950)

[3.5. Operating Environment: 24](#_Toc102219951)

[3.6. Design and Implementation Constraints: 24](#_Toc102219952)

[3.7. User Documentation: 24](#_Toc102219953)

[3.8. Assumptions and dependencies: 24](#_Toc102219954)

[3.9. External Requirement: 25](#_Toc102219955)

[3.9.1. Interface for user: 25](#_Toc102219956)

[3.9.3. Interface for backend: 25](#_Toc102219957)

[3.9.3. Interface for Software: 25](#_Toc102219958)

[3.10. Functional Requirements: 25](#_Toc102219959)

[3.10.1. R1: User Registration 25](#_Toc102219960)

[3.10.2. R2: Metamask Wallet Sign in 25](#_Toc102219961)

[3.10.3. R3: NFT minting 26](#_Toc102219962)

[3.10.4. R4: NFT Auctioning 26](#_Toc102219963)

[3.10.5. R5: NFT Bidding 26](#_Toc102219964)

[3.10.6. R6: Bid Acceptance 26](#_Toc102219965)

[3.10.7. R7: NFT Buying 26](#_Toc102219966)

[3.11. Non-Functional Requirements: 26](#_Toc102219967)

[3.11.1. Non-Functional Requirement for Frontend: 26](#_Toc102219968)

[3.11.2. Non-Functional Requirements for Smart Contracts: 27](#_Toc102219969)

[3.12. Use Case: 28](#_Toc102219970)

[3.12.1. Use Case Diagram: 28](#_Toc102219971)

[3.12.2. Use Case Description: 29](#_Toc102219972)

[3.13. Sequence Diagram: 34](#_Toc102219973)

[3.13.1. Register: 34](#_Toc102219974)

[3.13.2. Sign-in: 35](#_Toc102219975)

[3.13.3. NFT-creation: 35](#_Toc102219976)

[3.13.4. NFT-Auction: 36](#_Toc102219977)

[3.13.5. NFT-Bidding: 36](#_Toc102219978)

[3.13.6. Following Creators: 37](#_Toc102219979)

[3.13.7. NFT Rating: 37](#_Toc102219980)

[**System Design** 38](#_Toc102219981)

[4.1. Design Approach: 39](#_Toc102219982)

[4.2. Design Constraints: 39](#_Toc102219983)

[4.2.1. Programming Languages: 39](#_Toc102219984)

[4.2.2. Design Standard: 39](#_Toc102219985)

[4.2.3. Assumptions and Dependencies: 39](#_Toc102219986)

[4.3. Class Diagram: 40](#_Toc102219987)

[4.4. ER Diagram: 41](#_Toc102219988)

[4.5. Attributes: 41](#_Toc102219989)

[4.6. Domain Constraints: 42](#_Toc102219990)

[**System Implementation** 44](#_Toc102219991)

[5.1. System Architecture: 45](#_Toc102219992)

[5.1.1. Design Overview: 45](#_Toc102219993)

[5.2. Tools Used: 45](#_Toc102219994)

[5.2.1. Remix IDE: 45](#_Toc102219995)

[5.2.2. VS Code Text Editor: 45](#_Toc102219996)

[5.2.3. Metamask Wallet: 46](#_Toc102219997)

[5.2.4. Pinata IPFS: 46](#_Toc102219998)

[5.2.5. Infura API: 46](#_Toc102219999)

[5.2.6. MongoDB: 46](#_Toc102220000)

[5.3. Languages Used: 46](#_Toc102220001)

[5.3.1. Solidity: 46](#_Toc102220002)

[5.3.2. JavaScript: 46](#_Toc102220003)

[5.3.3. HTML: 46](#_Toc102220004)

[5.3.4. CSS: 46](#_Toc102220005)

[5.4. Methodology: 47](#_Toc102220006)

[5.4.1. User Services: 47](#_Toc102220007)

[5.4.2. Creators Services: 47](#_Toc102220008)

[**System Testing and Evaluation** 48](#_Toc102220009)

[6.1. GUI Testing: 49](#_Toc102220010)

[6.2. Usability Testing: 49](#_Toc102220011)

[6.3. Integration Testing: 49](#_Toc102220012)

[6.4. Exception Handling: 49](#_Toc102220013)

[6.5. Smart Contract Testing: 49](#_Toc102220014)

[6.5. Test Cases: 50](#_Toc102220015)

[**Conclusion** 55](#_Toc102220016)

[7.1. Future Improvements: 56](#_Toc102220017)

[7.1.1. Integrated DEX: 56](#_Toc102220018)

[7.1.2. Staking Module: 56](#_Toc102220019)

[7.1.3. Interoperability: 56](#_Toc102220020)

[***References:*** 57](#_Toc102220021)

# 

# **Certificate**

The work in the document “NFT Marketplace” is written by Fraz Naveed as a confirmation of the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Mr. Ali Irfan (Assistant Professor)

Internal Examiner: Name of the Internal Examiner (Title)

External Examiner: Name of the External Examiner (Title)

Project Coordinator: Dr. Muazzam (Sr.Assistant Professor)

Head of the Department: Dr. Faisal Bashir (Associate Professor)

9th Nov 2021

# 

# **Abstract**

With the boom of blockchain development, many use cases arose with it. One of the most innovative concepts it brought with it is Non-Fungible Tokens. The blockchain is a distributed ledger that is immutable and provides the certificate of authenticity of anything that is stored on it. NFTs are the assets that can be particularly associated with anything digital or physical. These NFTs are also a currency but are not interchangeable like fungible tokens which can be traded with each other. Every NFT is a unique asset, has unique features, and, holds its value. The making of an NFT marketplace will provide an opportunity to buy and sell off those unique assets. This project is unique in the sense that it will allow gaming content creators to showcase their best gaming moments as an NFT and sell its rights to their fans. When a thing is unique it holds significant value, applying that concept the NFTs then could be sold further offering more capital to both buyer and the seller.

# 

# **Acknowledgments**

I, first of all, thank Almighty ALLAH (SWT) for giving us the knowledge, power, and, strength to accomplish this project. I owe a great many thanks to my friends and the people who helped and supported my idea and worked with me on this project. My sincere gratitude to Mr. Ali Irfan (Assistant Professor in the Department of Computer Science, Bahria University) for having faith in me and thus allowing me to carry out a project. He helped immensely by guiding me throughout the project, inspiring me to take up new challenges along the road, and at the same time providing valuable suggestions and constructive criticisms. Without him, this project might not have been completed.

Fraz Naveed

Islamabad, Pakistan

November 9, 2021

# 

# **Table of figures**

[Figure 1. Project Lifecycle 12](#_Toc93882289)

[Figure 2. Project Architecture 13](#_Toc93882290)

[Figure 3. Activity Diagram 22](#_Toc93882291)

[Figure 4. Use Case diagram 26](#_Toc93882292)

[Figure 5. Class Diagram 38](#_Toc93882293)

[Figure 6. ER Diagram 39](#_Toc93882294)

# **Table of tables**

[Table 1. Milestones 17](#_Toc102213992)

[Table 2. Document Terms 22](#_Toc102213993)

[Table 3. Software Interface 25](#_Toc102213994)

[Table 4. Use Case-Register 28](#_Toc102213995)

[Table 5. Use Case-Sign in 28](#_Toc102213996)

[Table 6. Use Case-NFT creation 29](#_Toc102213997)

[Table 7. Use Case- NFT Auctioning 30](#_Toc102213998)

[Table 8. Use Case- NFT Bidding 30](#_Toc102213999)

[Table 9. Use Case- Following Creators 31](#_Toc102214000)

[Table 10. Use Case-NFT Rating 31](#_Toc102214001)

[Table 11. Attributes 40](#_Toc102214002)

[Table 12. Domain Constraints-User 41](#_Toc102214003)

[Table 13. Domain Constraints-Creator 41](#_Toc102214004)

[Table 14. Domain Constraint-Rating 41](#_Toc102214005)

[Table 15. Domain Constraints-Payment 42](#_Toc102214006)

[Table 16. Test Case-Login 49](#_Toc102214007)

[Table 17. Test Case-Register 49](#_Toc102214008)

[Table 18. Test Case- NFT creation 50](#_Toc102214009)

[Table 19. Test Case- NFT Auctioning 50](#_Toc102214010)

[Table 20. Test Case- NFT Bidding 51](#_Toc102214011)

[Table 21. Test Case- Bid Acceptance and Transfer 51](#_Toc102214012)

[Table 22. Test Case- Following Creators 52](#_Toc102214013)

[Table 23. Test Case- Unfollowing Creator 52](#_Toc102214014)

[Table 24. Test Case- NFT Upvoting 53](#_Toc102214015)

[Table 25. Test Case- NFT Downvoting 53](#_Toc102214016)

# **Acronyms and abbreviations:**

UC Use Case

NFT Non-Fungible Token

DAPP Decentralized Application

ABI Application Binary Interface

API Application Programming Interface

IDE Integrated Development Environment

JS JavaScript

RPC Remote Procedure Calls

DEFI Decentralized Finance

IPFS Interplanetary File System

ERC Ethereum Request for Comment

**Chapter # 1**

# **Introduction**

## Introduction:

An NFT marketplace will be developed that will be connected with the Ethereum blockchain, on which creators can post their gaming moments as NFTs and, buyers can buy and bid on those NFTs. NFTs are non-fungible tokens, with each token having unique properties. There will be a native token for the platform that will be listed on a DEX (Decentralized Exchange) which users will use for transactions on the platform. The NFTs will be supported by the ERC721 standard with the additional auction smart contract for the auctioning and bidding of the NFTs.

### Creator Perspective:

The content creator will use this platform to mint the NFTs. These are the things he can do:

#### **Minting:**

This feature is the core feature of the platform. With the minting functionality, the creator will craft the asset and it will be sent to be stored on the Ethereum blockchain.

#### **Auctioning:**

Although all the NFTs will get minted with their price this auction feature will allow the creators to put limited edition pieces of NFTs on sale for a limited time.

### Buyer Perspective:

* + - The buyer will be able to see the NFTs posted on the platform
    - He can bid on those NFTs.
    - He can search and sort the NFTs on the platform
    - He can follow his favorite creator.

## Objective:

The objective is “*To design an NFT marketplace for unique gaming moments that will live on Ethereum blockchain with the buy, sell and auction mechanics*”.

## Problem Description:

An NFT marketplace is a place for buying and selling digital assets. These marketplaces are used to create, display, and trading of digital assets using tokens of the native platform or blockchain. Before the concept of NFTs, the creators would create their physical artworks and place them in galleries for sale. But that was the case only for picture arts, for the displaying and selling of video artworks these digital markets can be used where any type of art can be sold and the ownership of the respective artwork could be preserved. On our marketplace, the content creators can post their gaming moments as NFT. Because gaming moments are themselves unique and those moments are unlikely to happen again they are perfect for the scenario of non-fungible tokens. The marketplace can be combined with blockchain, the art that will be created by the user will get stored on the blockchain, and from there it will be displayed on the platform. The origin of the artwork on these platforms is public and anyone can verify it.

## Methodology:

For the making of this project, the agile methodology will be used. Agile methodology is an iterative process in which changes are released with improvements. Right now, the plan for the project is limited and new features could be added later on. But the contracts that are already deployed will not be subject to change. For the integration of new features, new smart contracts will be developed in which addresses of the previously deployed contracts will be used for interaction purposes. For the development of contracts, the standard approach will be followed that is provided by the OpenZeppelin.

The frontend website will be made on React.js and Web3.js library will be used for the interaction with the blockchain. Pinata SDK will be used for the dumping of the digital assets, which will be done with the help of multer and the critical data will be stored on the blockchain. For the storage of data of users, a database like MongoDB will be used.

#### Lifecycle**:**

A system will be created that will consist of a web application and code will reside on the Ethereum blockchain. The system will be designed and built with solidity language, tested on Metamask wallet and truffle, deployed on Ethereum blockchain, and monitored from Etherscan.

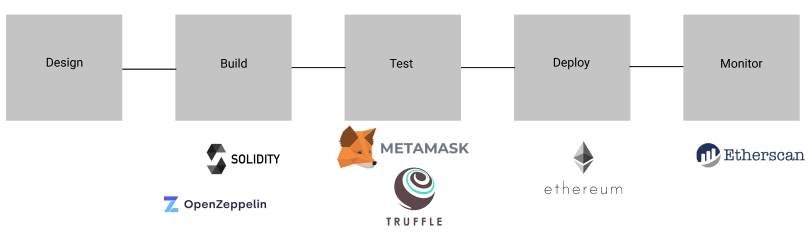


Figure . Project Lifecycle

### Architecture:

* Presentation Layer
* Business Layer
* Blockchain Layer

##### Presentation Layer:

The Presentation Layer will provide the following functionalities:

* Wallet connection
* Signup form
* Minting the NFTs
* Viewing other NFTs
* Bidding on NFTs
* Following other creators

##### Business Layer:

The Business Layer will do the following:

* Handle requests from the frontend
* Forward requests to blockchain
* Handle MongoDB logic
* Handle dumping of NFTs to IPFS

##### Blockchain Layer:

The Blockchain Layer will do the following:

* Hold our smart contracts
* Save relevant data coming from the Business Layer
* Transfer of asset’s logic

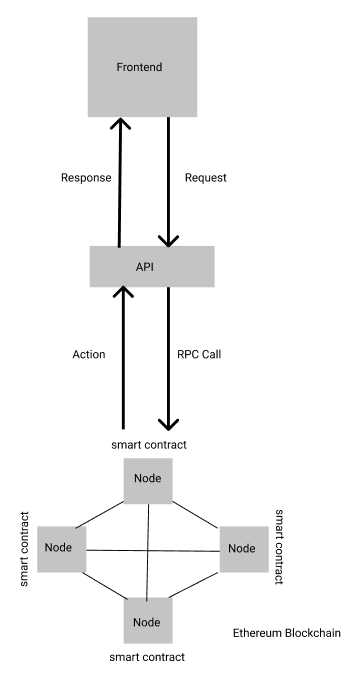


Figure . Project Architecture

## Project Scope:

### Justification:

A platform is needed where content creators could post their unique gaming moments as NFT and that could be sold online along with the ownership protection.

### Deliverables:

In this project, an NFT marketplace will be built. The creators will be able to sign up on the platform, the data of whom will be stored on a separate centralized database. The user will buy the platform’s native tokens that will be developed (which will be given some name in the future) and with the help of it, he will be able to mint their artwork. He will specify the particular asset’s price, and that asset will get stored on the blockchain. From blockchain that asset will be shown to other users on the frontend. The creator will have additional functionality of putting that asset on auction. Other users will be able to bid on those assets if they are on auction. If the original creator liked the bid, he will stop the auction and the NFT will get transferred to the highest bidder and the creator will get that particular number of tokens. An additional fund could be allocated for the maintenance of the platform i.e., whenever there is some sale a portion of it goes to developers for maintenance purposes.

For the token, NFT and auction mechanics smart contracts will be made and for testing, they will be deployed on the Ropsten Testnet of Ethereum.

### Exclusion:

Upgradability feature of smart contracts that could override the existing smart contracts. We cannot change the smart contracts once they are deployed on the blockchain. In case a bug arises in the future it could cause trouble. So, this approach is used in such cases. This approach is complex yet very powerful in terms of the management of smart contracts. This feature won’t be included right now.

## Feasibility Study:

### Risks Involved:

#### **For project:**

Scope Creep:The project might seem small at the start and adding more and more functionality to it can make it out of scope.

Scheduling:Time management issues could arise if the scope gets larger and larger.

Integration Risk:Due to the complexity of the project while integrating, some high severity bug could make their way into the platform as a single individual is looking after all processes.

Legal Risk: Full-scale launch of such a project could invite trouble from LEAs as crypto isn’t regulated yet.

#### **For platform:**

Asset ownership risks: Risk of theft or loss of digital assets where someone puts assets claiming to be theirs.

Smart Contract security risk: If smart contracts are not tested and audited before deployment, they could pose a serious issue for the theft of assets or tokens.

### Resource Requirement:

Human Resource for testing: Because the project is complex and critical, as real money is involved, a certified organization should audit smart contracts.

Capital Resource: If the project is launched fully, then cost resources would be required to cover the expenses for the hosting of the platform and deployment of smart contracts on the main Ethereum network.

## Solution Application Areas:

The project has value in the Art markets present in the digital space. The NFT marketplaces take the concept of ownership of art to another level where the ownership of the respective item is preserved on the temper-proof blockchain.

By putting it on the marketplace combined with blockchain, everyone agrees upon the fact that the particular item was created by the specific creator and he is the owner of that item.

## Tools/Technology:

* Solidity
* Vanilla JavaScript
* Web3.js
* Metamask
* ERC20, ERC721 tokens
* Pinata SDK
* Infura
* Multer
* React.js
* MongoDB
* Express.js
* Node.js

## The expertise of the Team Members:

The team has a deep understanding of how the Ethereum blockchain protocol works. Along with that team is fluent in Solidity Programming language, libraries of blockchain, designing of frontends, and connecting them.

## Milestones:

Table . Milestones

|  |  |
| --- | --- |
| **Milestones** | **Expected Completion** |
| Project Proposal submission | Oct 20, 2021 |
| Proposal Defense | Oct 29, 2021 |
| Frontend Design | Nov 15, 2021 |
| Documentation Phase-1 | Jan 10, 2022 |
| Implementation Phase-1 | Jan 10, 2022 |
| Documentation Phase-2 | Mar 15, 2022 |
| Implementation Phase-2 | Mar 15, 2022 |
| Final Review & Submission | Apr 25, 2022 |

**Chapter # 2**

# **Literature Review**

## 2.1. Literature Review:

## 

NFT marketplace will provide the opportunity for creators to post their gaming content as Non-Fungible Tokens on a web-based application. Posting their rare moments as an NFT not only prevents it from being stolen but also preserves the ownership of their content. This content can help the creators earn money by selling them to other users.

NFTs had been a buzzword this year with lots of new concepts of them appearing with each passing day. Much of the NFT-related work has been done on the Ethereum blockchain that targets mostly JPEG/PNG-based NFTs. The creation of this project has been aimed with creator royalties in mind. Following is some discussion related to NFTs and the work that had been done.

### 2.1.1. Insight on NFTs and the work done:

The concepts of Defi and Non-Fungible Tokens are among those new concepts in blockchain that have been recently introduced to the market. In the “Non-Fungible Tokens”, the word non-Fungible means the indivisible assets which have their unique properties and cannot be exchanged with other assets. Each Non-Fungible asset is unique in the sense that each one holds its distinctive properties and has its value. They are the non-interchangeable unit of data and cannot be replaced with anything else. To simplify this, we can take an example of a house. Each house has its unique attributes and holds its price tag according to the location and the built quality. Any other house cannot be exchanged with that house because of the uniqueness of their own. This notion of non-Fungibility is applied to tokens. Tokens in the blockchain are something that a person is capable to own. So Non-Fungible Tokens are the assets on the blockchain which a person can own with their ownership record of it publicly shown.

With many other UCs of Non-Fungible tokens, there is one UC that targets the gaming industry. The gaming industry has exploded in recent years with lots of revenue potential. Gamers compete with each other to emerge as one of the best in the whole world. While they compete with each other, they produce intense and nail-biting moments. Each gaming moment is rare in the sense that those moments cannot be copied and are unlikely to happen again. Those extreme moments with their uniqueness can serve the purpose of non-Fungible tokens. The creators can make use of those moments and sell them as non-Fungible tokens to their fans as digital collectibles.

One of the prime examples of the current NFT marketplace is OpenSea. It’s a marketplace that targets mostly JPEG/PNG-based NFTs. This marketplace has Ethereum blockchain on its backend with the frontend and logic written in JavaScript and TypeScript.

“NFT Marketplace” is a project that is similar to online shopping stores. But the difference is that this marketplace intends to sell digital items. With the item sold its record gets stored on the blockchain and everyone can see on public portals who owns this NFT.

The concept behind the project is to allow creators to sell their gaming content as tokenized video clips of their intense gaming moments. The marketplace has its coin that will be used for all the transactions happening on the marketplace. The marketplace will be like a social place where fans can follow their favorite creators. The creators will earn royalties when there is a change of hands in NFTs. A part of those royalties will be reserved for future projects in the marketplace.

### 2.1.2. Conclusion:

This chapter had been a subject for giving insights on what exactly are NFTs, what is the already implemented marketplace, and what I am doing in my marketplace. This all has helped in getting crucial concepts, work done, and how it can be used in future work for my project.

**Chapter # 3**

# **Requirement Specification**

## 3.1. Document Convention:

The following terms are used in the document:

Table . Document Terms

|  |  |
| --- | --- |
| AD | Architecture Diagram |
| ER | Entity Relationship |
| UC | Use case diagram |
| SD | Sequence Diagram |
| SD | State Diagram |
| CD | Class diagram |
| CD | Component Diagram |
| ACTD | Activity Diagram |

## 3.2. Intended Audience:

The project “NFT Marketplace” will be a project intended for the international gaming community in particular. First, it will be deployed on the testnet of the Ethereum blockchain. But when contracts are audited and tested for security by concerned auditors, they will be ready to be fully launched on the Ethereum mainnet.

## 3.3. Marketplace Activities:

The main activities of the project are shown in the activity diagram:

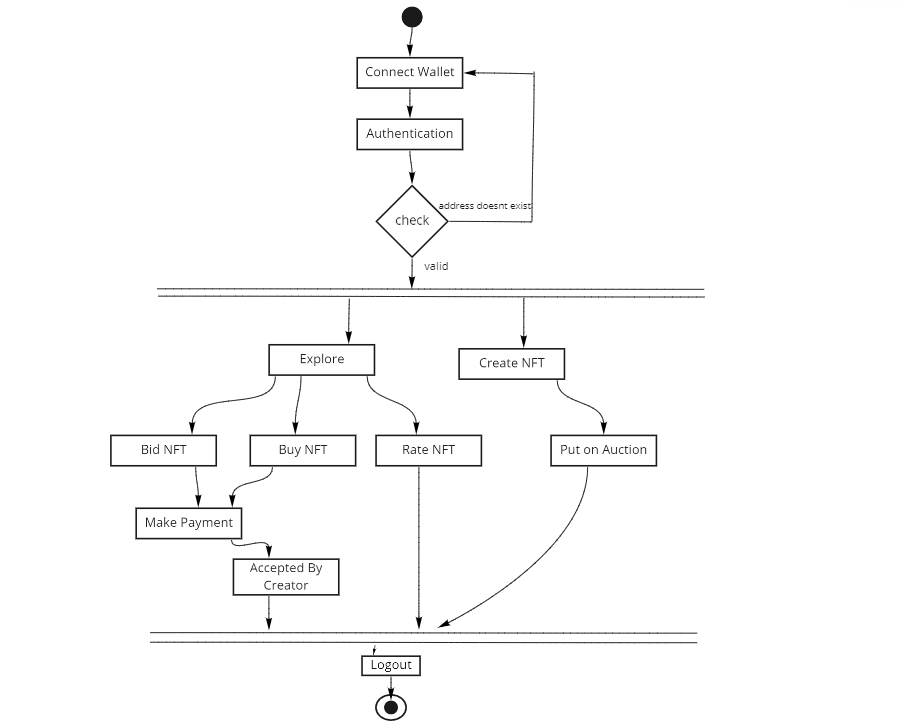


Figure . Activity Diagram

## 3.4. User Characteristics and class:

Whenever a user will come to the marketplace, he will be prompted to connect the wallet to do the activity on the platform. Otherwise, he will only be just able to explore the NFTs. When the user will connect to the wallet, the system will check in the database whether the address of the user exists or not. If it doesn’t exist then he will be prompted to the signup form. If the user already exists then he can normally surf the marketplace and perform activities.

The creators can put their NFT on the marketplace by minting it through the create NFT form. As soon as the transaction is confirmed he will be able to see the uploaded NFT on the marketplace. He then can put the NFT in an auction if he wants. For the normal user, he will be able to explore through the marketplace using filters and search functionality. If he likes any NFT then he can give a rating to it. Rating will give insight to other users on how trending and valuable NFT is. If the user likes any NFT from the auction section, he can place his bids on that NFT which should be greater than the previous one. Meanwhile, his ERC20 tokens will get transferred to a smart contract equal to the amount of bid he placed. If the creator likes the bid, he can transfer that NFT to the last bidder and the ERC20 tokens locked in the smart contract will get transferred to the creator.

Users can perform the following functionalities:

### 3.4.1. User Functionalities:

* + - * Signup
      * Login
      * Bidding
      * Buying
      * Rating
      * Following

### 3.4.2. Creator Functionalities:

* + - * Signup
      * Login
      * Creating NFT
      * Auctioning
      * Buying
      * Rating
      * Following

## 3.5. Operating Environment:

The operating environment for the NFT marketplace is:

* Frontend: React.js
* API: node.js/Web3.js
* Blockchain: Ethereum

## 3.6. Design and Implementation Constraints:

* RPC calls will be performed from API to blockchain
* API will be hosted on centralized, so this application is a hybrid decentralized

## 3.7. User Documentation:

The primary goal of this NFT marketplace is to make it as user-friendly as possible. The application will be designed in such a way that the user platform keeps telling the user what is happening. Despite all of this, as the NFT field is relatively new to the users' documentation will be still required to get in-depth insights into the concepts and working of the application.

## 3.8. Assumptions and dependencies:

* Smart contracts are depending on each other. The auction Contract will be depending upon the ERC721 contract and the ERC721 contract will be depending upon the ERC20 contract. So, the values in each contract have to be updated.
* The smart contracts are designed in such a way that once the user bids, his ERC20 tokens will get deducted from his balance. So, he has to think before bidding.

## 3.9. External Requirement:

### 3.9.1. Interface for user:

Frontend: React.js, CSS

### 3.9.3. Interface for backend:

Backend: Node.js, Web3.js

### 3.9.3. Interface for Software:

Table . Software Interface

|  |  |
| --- | --- |
| Software | Description |
| OS | The application will run on a browser without depending upon any specific OS. |
| Blockchain | Ethereum blockchain will be used for smart contracts. |
| React.js | React.js is chosen tool for front-end development. |

### 

## 3.10. Functional Requirements:

### 3.10.1. R1: User Registration

**Description**: Address of the Metamask wallet, Username.

**Input**: User details.

**Output**: Filled Registration details.

**Processing**: The user’s address will be checked with the database.

### 3.10.2. R2: Metamask Wallet Sign in

**Description**: Wallet should prompt automatically when a user comes to the website.

**Input**: User password.

**Output**: Sign-in of a user on the platform.

**Processing**: Metamask wallet will authenticate the user password.

### 3.10.3. R3: NFT minting

**Description**: NFT should be uploaded and shown on the marketplace.

**Input**: NFT metadata, title, description.

**Output**: NFT is shown on the platform.

**Processing**: The NFT will be uploaded on IPFS and IPFS hash gets stored on a smart contract.

### 3.10.4. R4: NFT Auctioning

**Description**: NFT status should get updated and shown in the auction section.

**Input**: Auction start date, end date, price.

**Output**: NFT gets shown in the auction section.

**Processing**: The call to put NFT on auction will be given to the smart contract.

### 3.10.5. R5: NFT Bidding

**Description**: Upon bidding on the NFT the latest bid should be shown on the frontend.

**Input**: Auction start date, end date, price.

**Output**: Bid on NFT updated.

**Processing**: The call to place a bid will be given to a smart contract which will update the bidding price.

### 3.10.6. R6: Bid Acceptance

**Description**: Upon accepting the bid by the creator the NFT should get transferred to the last bidder and the User balance in the frontend be updated.

**Input**: Acceptance of bid.

**Output**: NFT ownership gets changed.

**Processing**: The call of acceptance of the bid will be given to the smart contract which will transfer the tokens and NFT to the user and creator.

### 3.10.7. R7: NFT Buying

**Description**: Buying of NFT transfers the tokens to the NFT owner and ownership of NFT is updated.

**Input**: Price of the NFT.

**Output**: NFT ownership updated.

**Processing**: The call to buy NFT will be given to a smart contract which in turn will transfer the tokens to the creator and NFT to the user.

## 3.11. Non-Functional Requirements:

### 3.11.1. Non-Functional Requirement for Frontend:

* When the user comes on the platform he should be prompted to connect to a wallet or he should be told to connect to the wallet.
* Whenever there is a minting call to the blockchain, there is a wait time as the transaction takes time in getting confirmed. So user should be shown a loading screen to keep it engaged.
* The user should be shown the provenance of the token on the frontend.
* There should be filters and search features available for the user.
* A dark & light theme toggle should be implemented.

### 3.11.2. Non-Functional Requirements for Smart Contracts:

* The smart contracts should be thoroughly audited for bugs before deploying on the mainnet.
* The code should be under the OpenZeppelin standards.

## 3.12. Use Case:

### 3.12.1. Use Case Diagram:

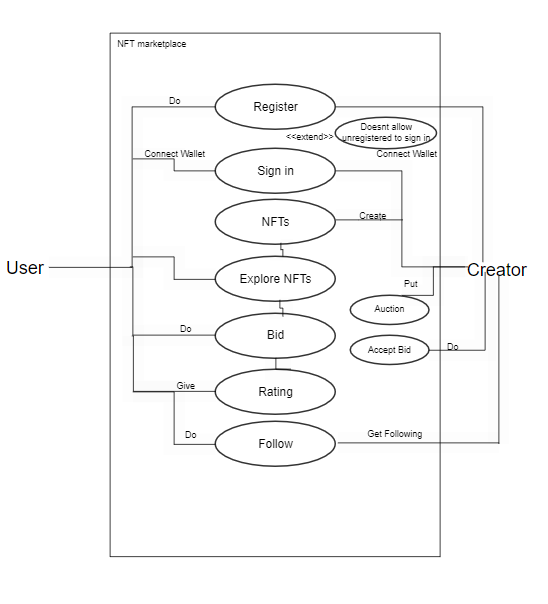


Figure . Use Case diagram

### 3.12.2. Use Case Description:

Table . Use Case-Register

|  |  |  |
| --- | --- | --- |
| Name of use-case | Register | |
| Use case ID | NM-BU01 | |
| Priority: | High | |
| Primary actor: | Creator, User | |
| Description: | This use case describes the registration of any user or creator that is coming to the platform. The user’s wallet address will be used to identify them and the username and profile picture would be used to show them on the marketplace. Giving these pieces of information will allow the user to use all the options available on the marketplace. | |
| Basic flow: | * The user presses the “connect wallet” button. * The relevant address gets searched on the database. * If the user is not found, his data is stored in the database. | |
| Pre-condition: | The user is not registered to our application. | |
| Trigger: | The use case is triggered when a new user comes to our platform. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The user connects the wallet and gives the relevant information. | Step 2: The information is checked on the database.  Step 3: If information is not already available, it gets stored in the database.  Step 4: The user is given access to the marketplace |
| Post-condition: | The account has been created and the user starts using the marketplace. | |

Table . Use Case-Sign in

|  |  |  |
| --- | --- | --- |
| Name of use-case | Sign in | |
| Use case ID | NM-BU02 | |
| Priority: | High | |
| Primary actor: | Creator, User | |
| Description: | This use case describes the sign-in of any user or creator that is coming to the platform. The user’s wallet address will be used to identify them and the username would be used to show them on the marketplace. | |
| Basic flow: | * The user presses the “connect wallet” button. * The relevant address gets searched on the database. * If the user is found he is given access to the marketplace. | |
| Alternative flow: | **User is not on Ethereum relevant network:**   * The user presses the “connect wallet button”. * The user is on another network. * Users won’t be able to use the marketplace. | |
| Pre-condition: | The user is already registered on the platform. | |
| Trigger: | The use case is triggered when an existing user comes to our platform. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The user connects the wallet and gives the relevant information. | Step 2: The information is checked on the database.  Step 3: If information is already available he is given access to the marketplace. |
| Post-condition: | The account has been signed-in and the user starts using the marketplace. | |

Table . Use Case-NFT creation

|  |  |  |
| --- | --- | --- |
| Name of use-case | NFT creation | |
| Use case ID | NM-BU03 | |
| Priority: | High | |
| Primary actor: | Creator | |
| Description: | This use case describes that the creator on the platform mints an NFT. | |
| Basic flow: | * The user selects the metadata for the NFT. * The user gives the title, price, and description for the NFT. * Upon clicking mint the NFT gets uploaded * NFT gets shown on the platform. | |
| Pre-condition: | Users should have enough gas fees to pay for a transaction. | |
| Trigger: | The use case is triggered when the creator creates an NFT. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The user gives the title, category, price, description, and metadata for the NFT. | Step 2: The NFT information is sent to the API.  Step 3: API uploads the image on IPFS  Step 4: IPFS returns a hash that API uploads in the smart contract.  Step5. Latest NFTs are fetched from the blockchain. |
| Post-condition: | The NFT has been uploaded and been showing on the platform. | |

Table . Use Case- NFT Auctioning

|  |  |  |
| --- | --- | --- |
| Name of use-case | NFT Auctioning | |
| Use case ID | NM-BU04 | |
| Priority: | High | |
| Primary actor: | Creator | |
| Description: | This use case describes that the creator wants to put NFT on auction. | |
| Basic flow: | * The creator selects the NFT to be put on sale * The creators give data for the auction. * NFT gets shown in the auction category. | |
| Pre-condition: | The user should have enough gas fee to pay for the transaction and must own that token. | |
| Trigger: | The use case is triggered when the creator put NFT on auction. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The creator selects the NFT to be put on Auction.  Step 2: The creator gives the start date, end date, and auction price. | Step 3: The relevant information is sent to the API.  Step 4: API sends a call to the smart contract  Step 5: Smart Contract updates the status of NFT.  Step 6: NFTs on auction call are sent from API |
| Post-condition: | The NFT has been put on auction and is shown in the auction section. | |

Table . Use Case- NFT Bidding

|  |  |  |
| --- | --- | --- |
| Name of use-case | NFT Bidding | |
| Use case ID | NM-BU05 | |
| Priority: | High | |
| Primary actor: | User | |
| Description: | This use case describes that the user wants to place a bid on an NFT that is on Auction. | |
| Basic flow: | * The user places a bid higher than the current price of the NFT. | |
| Pre-condition: | Users should have enough gas fees to pay for the transaction and must have equivalent ERC20 tokens to that price. | |
| Trigger: | The use case is triggered when the user places a bid on NFT. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The user places a bid on the NFT and the bid price is higher than that of the previous price. | Step 2: The relevant information is sent to the API.  Step 3: API sends a call to the smart contract  Step 4: Smart Contract updates the price of NFT. |
| Post-condition: | The NFT bid price has been updated and is being shown on the marketplace. | |

Table . Use Case- Following Creators

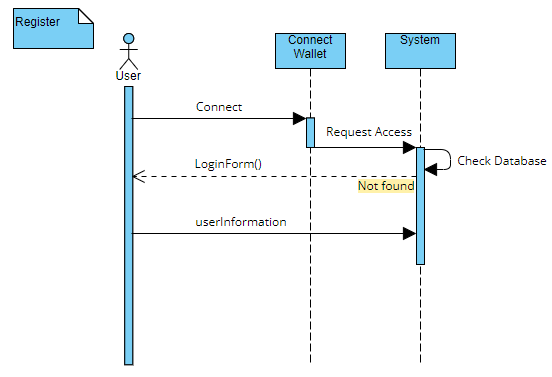
|  |  |  |
| --- | --- | --- |
| Name of use-case | Following Creators | |
| Use case ID | NM-BU06 | |
| Priority: | Low | |
| Primary actor: | Users, Creators | |
| Description: | This use case describes that the user and creators can follow other fellow creators. | |
| Basic flow: | * The follow button on the profile card is pressed. | |
| Pre-condition: | The user or Creator must be signed in on the platform. | |
| Trigger: | The use case is triggered when someone gets followed. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The creator or user presses the follow button that is present on the profiles. | Step 2: The increment is sent to the database.  Step 3: The number of followers is updated in the table of the database. |
| Post-condition: | The number of followers is increased. | |

Table . Use Case-NFT Rating

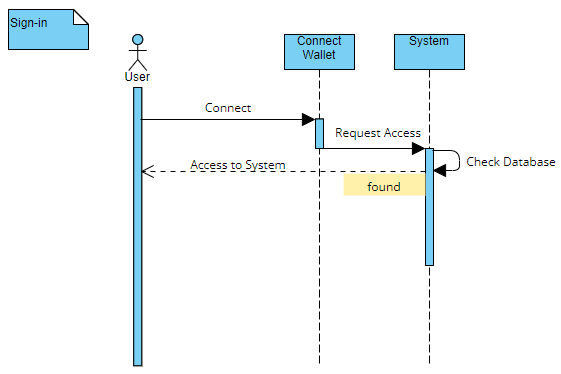
|  |  |  |
| --- | --- | --- |
| Name of use-case | NFT Rating | |
| Use case ID | NM-BU07 | |
| Priority: | Low | |
| Primary actor: | Users, Creator | |
| Description: | This use case describes that the rating of NFT is increased. | |
| Basic flow: | * The creator or user presses the upvote buttons on the NFT card. | |
| Pre-condition: | The user or creator must be signed in on the platform. | |
| Trigger: | The use case is triggered when the NFT rating is changed. | |
| Course Event: | **Actor Action** | **System Response** |
|  | Step 1: The creator or user gives stars to the NFT. | Step 2: The change in rating is sent to the database.  Step 3: The rating of the relevant NFT is updated. |
| Post-condition: | A new rating of the NFT is shown on the platform. | |

## 3.13. Sequence Diagram:

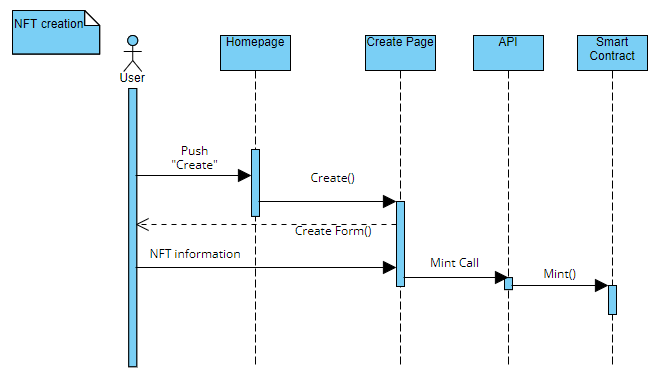
### 3.13.1. Register:



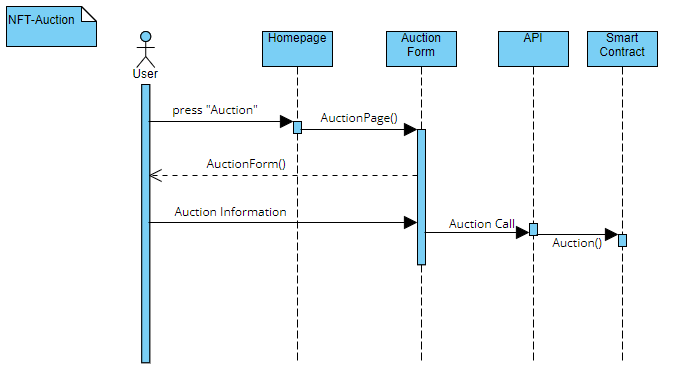
### 3.13.2. Sign-in:



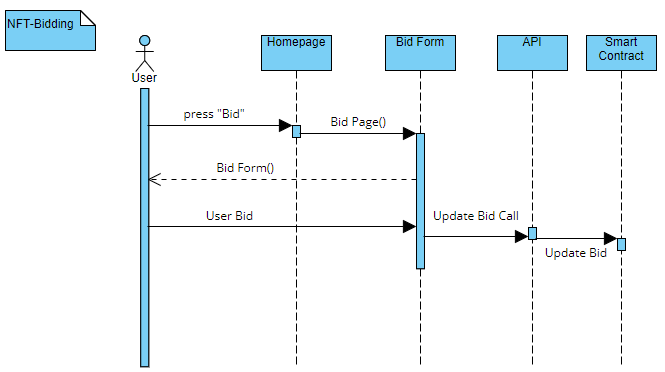
### 3.13.3. NFT-creation:



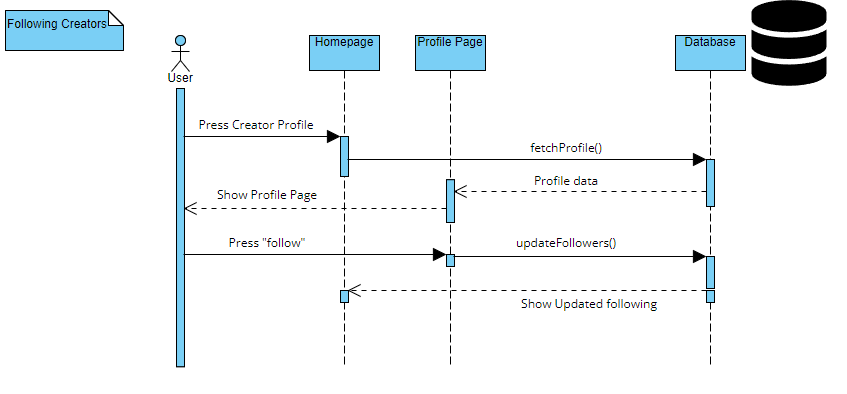
### 3.13.4. NFT-Auction:



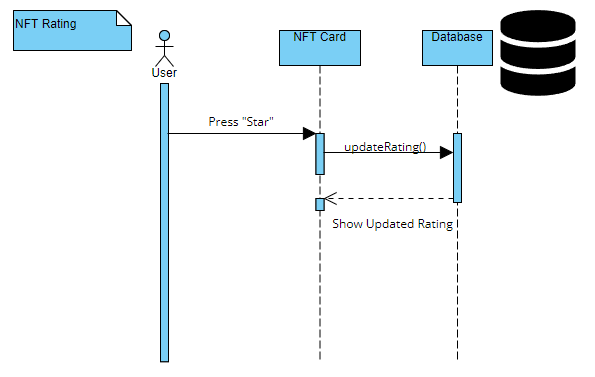
### 3.13.5. NFT-Bidding:



### 3.13.6. Following Creators:



### 3.13.7. NFT Rating:



**Chapter # 4**

# **System Design**

## 4.1. Design Approach:

As Solidity language is being used in designing the smart contracts which are powering the data layer, a new approach called Contract-Oriented Programming is being used to develop them. The new approach is a subset of Object-Oriented Programming but with new concepts of data structures that help in maintaining the funds and addresses at the smart contract level. The contracts provide the same data hiding techniques but also gives view and pure features for each of the function.

## 4.2. Design Constraints:

### 4.2.1. Programming Languages:

The main programming language being used in the project is **Solidity** which is being used for developing smart contracts. API is being developed in **node.js** along with the usage of **IPFS**, **web3.js**, and **MongoDB**. With the frontend being developed with the help of **React.js**.

### 4.2.2. Design Standard:

The design approach behind this project is 3-tier architecture. In a 3-tier architecture, the presentation layer is separated from the data layer. In the middle, there is a business logic layer. All the relevant calls from the frontend are handled at the business layers and then it is forwarded to the data layer which in our case is blockchain. I am using 3-tier architecture in this project to achieve decoupling of all the logic. The logic of RPC calls to the blockchain can be implemented in the frontend too, but this can complicate our codebase. So, a 3-tier architecture is the best solution with hybrid decentralization.

### 4.2.3. Assumptions and Dependencies:

* + - The source code will be available on the GitHub repository for anyone to view.
    - The API has to be kept alive 24/7 as all the important calls are being sent from this.
    - The codebase of smart contracts should be free of errors before launching on the mainnet.
    - As our main thing in this project are smart contracts, the MongoDB server getting down shouldn’t be the problem as non-important data is being stored there.
    - With the royalty fee from NFTs, more space on IPFS will be bought for the hosting of NFT metadata when the user base gets larger.

## 4.3. Class Diagram:

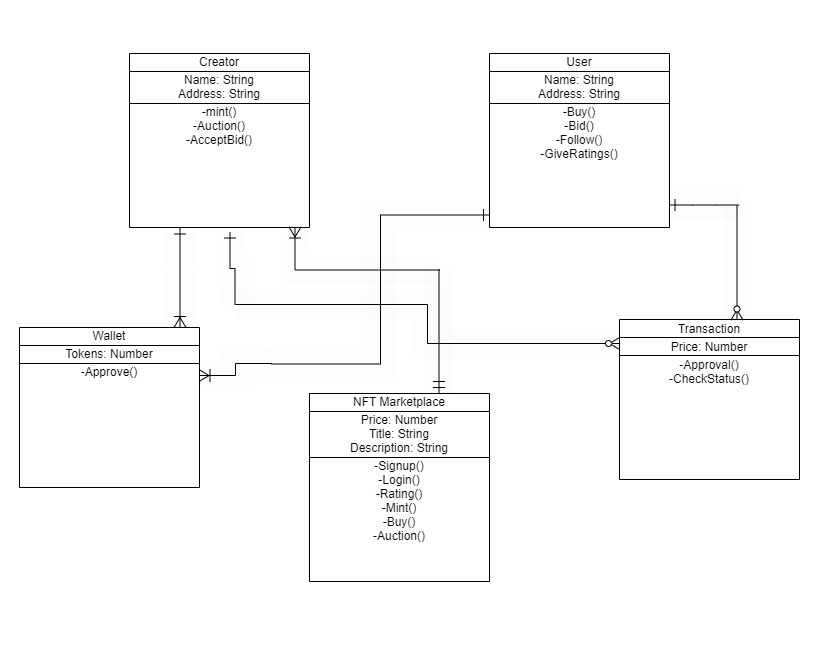


Figure . Class Diagram

## 4.4. ER Diagram:

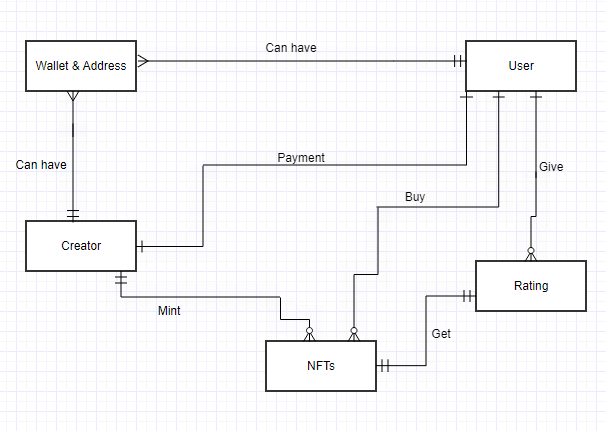


Figure . ER Diagram

## 4.5. Attributes:

Table . Attributes

|  |  |
| --- | --- |
| Wallet | * Address * Number of ERC20 tokens |
| User | * Wallet Address * Username * Profile Picture |
| Creator | * Wallet Address * Username * Profile Picture |
| NFTs | * Title * Description * Price * NFT metadata |
| NFT rating | * Number |

## 4.6. Domain Constraints:

**User**

Table . Domain Constraints-User

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Domain | Datatype | Size | Is Null | In unique | Default value | Range |
| Address | address | 42 | No | Yes | -- | -- |
| Profile Picture | Binary format file |  | No | No | -- | -- |
| Username | String | 15 | No | No | -- | -- |

**Creator:**

Table . Domain Constraints-Creator

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Domain | Datatype | Size | Is Null | In unique | Default value | Range |
| Address | address | 42 | No | Yes | -- | -- |
| Profile Picture | Binary format file |  | No | No | -- | -- |
| Username | String | 15 | No | No | -- | -- |

**Rating:**

Table . Domain Constraint-Rating

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Domain | Datatype | Size | Is Null | In unique | Default value | Range |
| Number | int | 5 | No | No | -- | -- |

**Payment:**

Table . Domain Constraints-Payment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Domain | Datatype | Size | Is Null | In unique | Default value | Range |
| Amount | Uint256 | 18 | No | No | 1 | >1 |
| Transaction status | Bytes | 32 | No | Yes | -- | -- |

**Chapter # 5**

# **System Implementation**

In this chapter, tools and techniques used for system implementation will be discussed.

## 5.1. System Architecture:

The system is divided into 3 modules:

1. Frontend
2. API
3. Blockchain

### 5.1.1. Design Overview:

Our application has been designed using the React.js library, where each page has been broken down into independent components. These components are then integrated to give the shape of the frontend. The frontend shows what the user can do with the application. Each page is self-explanatory and the user knows what to do on each page. The main part of the frontend is the NFT creation page. This is where NFT is created and sent to the blockchain. Each page has been connected with the backend API, which handles the requests coming from the frontend. The API has been designed using the Node.js framework. This API sends calls to Pinata IPFS, MongoDB database, and further on the blockchain. The IPFS and MongoDB have been used to assist with some problems in our application. For Example. The blockchain isn’t yet scaled at that much level to store images. To overcome the problem, NFT media has been dumped on the IPFS, the hash of which is then stored on the blockchain. Some data isn’t that much of important to be stored on the blockchain. As each thing on the blockchain has a cost, saving that data could raise our fees. So, for that purpose, MongoDB has been used. The API then further connects with the Ropsten testnet of the Ethereum blockchain, where it interacts with the deployed smart contracts. Requests are sent from the frontend to the API which sends them to the blockchain.

## 5.2. Tools Used:

Tools that helped in the completion of the project are:

### 5.2.1. Remix IDE:

The Remix IDE is an online text editor for writing, deploying, and administrating smart contracts for blockchains. All the smart contracts had been deployed through it.

### 5.2.2. VS Code Text Editor:

Visual Studio code has been used for writing the code for frontend and backend API. This text editor provided various functionalities e.g., ESlinting, and debugging for the development code.

### 5.2.3. Metamask Wallet:

Metamask wallet is a browser extension that holds our funds. All the transactions that have been signed through it.

### 5.2.4. Pinata IPFS:

Pinata IPFS is an InterPlanetary File System, which gives p2p storage of files. Not all data can be stored on the blockchain, so the NFT media was dumped on Pinata IPFS and their content identifiers (CID) were stored on the blockchain.

### 5.2.5. Infura API:

To connect with the blockchain, either we have to run our node, which is expensive in terms of storage, or we can connect with the providers that provide the connection to the blockchain. Infura API was used in the backend of our project to connect with the Ropsten testnet of the Ethereum blockchain.

### 5.2.6. MongoDB:

MongoDB database has been used to store the profile images, addresses, and relevant information to the profile.

## 5.3. Languages Used:

### 5.3.1. Solidity:

Solidity is an object-oriented language similar to C++ & JavaScript used for implementing smart contracts in EVM compatible blockchains. Solidity has been used for the writing of smart contracts in this project.

### 5.3.2. JavaScript:

JavaScript has been extensively used in this project. The frontend is used in the React.js library and the API has been developed in the node.js framework.

### 5.3.3. HTML:

HTML has been used for the designing of the frontend in the React.js library. It is used along with JavaScript for rendering the frontend pages.

### 5.3.4. CSS:

For the styling of pages, pure CSS has been used. The stylesheets are attached with HTML pages in React.js to style the components.

## 5.4. Methodology:

The system has 2 types of service providers that are available on the web application.

1. User Services
2. Creator Services

### 5.4.1. User Services:

The user services are designed for the new users that will first visit the application. They will connect with their Metamask wallet, first, they will be checked whether their wallet address exists in the database. If the user doesn’t exist then, they will be prompted to register themselves and that data will be saved on MongoDB. They will be then allowed to go through the marketplace. Each user can become a creator if they create an NFT on create page. Each user has the following options:

1. Explore the Marketplace
2. Create an NFT
3. Bid on the NFTs that are on auction.

### 5.4.2. Creators Services:

The creators have the same options for exploring the marketplace except they have an advantage over the normal users, that they can put their NFTs on auctions and take bids from the normal users. They can accept the bids and transfer the NFTs.

**Chapter # 6**

# **System Testing and Evaluation**

Testing is one of the most important parts of any decentralized application. Because smart contracts are once deployed on the blockchain are immutable. Any bug that exists in the smart contract can pose a risk to the funds and tokens stored on the smart contract. On the application side, testing is necessary to make sure the system is working as it is intended. The main purpose of testing is to remove as many flaws as possible and improve the system.

## 6.1. GUI Testing:

The GUI is the part of the application where the user will interact directly. The GUI has to be flawless to handle the requests of the user. The GUI must provide the user with some functionalities to properly explore the marketplace. Therefore, those functionalities like searching and sorting were thoroughly checked.

## 6.2. Usability Testing:

Usability is the part of the application that provides ease to the end-users in the exploration of the application. During usage of our application, there are some points where a user has to wait sometime during process handling at the backend. At this time user could lose interest. So those specific points are identified and proper feedback mechanisms like loaders have been implemented.

## 6.3. Integration Testing:

Our application is connected with smart contracts on the blockchain. They must be properly integrated with the frontend to provide the functionality to the end-user. Although users can interact with the smart contracts from Etherscan, that is not user-friendly. The GUI should be showing changes at the blockchain level in real-time. Those features were also tested properly.

## 6.4. Exception Handling:

The important part of GUI was to handle the connection of the Metamask wallet. If Metamask isn’t properly configured, it could cause the application to crash. The application was properly configured to handle the users that don’t have a Metamask wallet installed.

## 6.5. Smart Contract Testing:

Smart contracts are the most important part of any decentralized application. They have to be flawless before being deployed on the mainnet. For our application, all the error handling has been done at the smart contract level. “Assert” and “Revert” have been used throughout the smart contracts to avoid any potential loss of funds or make contracts go undefined. Furthermore, for the time being, they have been deployed on the Ropsten testnet and the application has been connected with it. Users will use the application and more flaws will be discovered which will help to eradicate the errors from the smart contract before being launched on the mainnet.

## 6.5. Test Cases:

Table . Test Case-Login

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User Login | T1 |
| **Description** | Test the login of the user from the Metamask wallet |  |
| **Requirement** | Internet connection, Metamask installed |  |
| **Initial Condition** | The application is online and the wallet connection page pop up; the User knows the password of their wallet. |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Enter password upon wallet prompt |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | If a user enters the correct password they are directed to the website. | Pass |
| **2** | If the user doesn’t enter the correct password in Metamask, they are stuck on connecting wallet button. | Pass |

Table . Test Case-Register

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User Registration | T2 |
| **Description** | User Registration if it’s their first-time o marketplace. |  |
| **Requirement** | Internet connection, Metamask installed |  |
| **Initial Condition** | The application is online the and user wallet is connected |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Enter password upon wallet prompt |  |
| **4** | Upload profile image and username |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | The user enters their details and they are saved in the database. | Pass |
| **2** | User doesn’t upload their details and press submit button; they are stuck on the current page. | Pass |

Table . Test Case- NFT creation

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | NFT creation | T3 |
| **Description** | The user tries to create the NFT on create page and its data is stored on the blockchain. |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and the user wallet is connected the and user has funds in their wallet to process the transaction. |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Click Create page |  |
| **4** | Enter NFT details |  |
| **5** | Press Mint button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | NFT details are uploaded to the blockchain. | Pass |
| **2** | The user doesn’t have enough funds in their wallet and NFT creation fails. | Pass |
| **3** | IPFS storage has run out of space for the user and NFT media isn’t uploaded on IPFS | Pass |

Table . Test Case- NFT Auctioning

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | NFT auction | T4 |
| **Description** | Creator puts NFT on auction |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and user wallet is connected and they have funds in their wallet to process the transaction. |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Go to the NFT details page |  |
| **4** | Enters bid price for NFT |  |
| **5** | Press Auction button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | NFT status on blockchain changes to Auction | Pass |
| **2** | The button is rendered to “Bid” instead of “Buy” the on NFT details page for normal users. | Pass |
| **3** | Put on bid status fails because the user didn’t have enough funds in their wallet. | Pass |

Table . Test Case- NFT Bidding

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | NFT bidding | T5 |
| **Description** | User bids on NFT that is on auction |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and user's wallet is connected and the user has funds in their wallet to process the transaction. |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Go to the NFT details page |  |
| **4** | Enters required bid price of NFT |  |
| **5** | Press the Make offer button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | NFT bid price is updated on the blockchain | Pass |
| **2** | Bid price tokens of the tokens are locked in the smart contract | Pass |

Table . Test Case- Bid Acceptance and Transfer

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | Creator accepts the bid | T6 |
| **Description** | Creator accepts the current bid on the NFT |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and user's wallet is connected and the user has funds in their wallet to process the transaction. |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Go to the NFT details page |  |
| **4** | Press Accept Bid button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | Bid tokens that are locked in a smart contract are transferred to the creator | Pass |
| **2** | NFT is transferred to the bidder | Pass |
| **3** | NFT is removed from the creator collection on GUI | Fail |

Table . Test Case- Following Creators

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User/Creator follow others | T7 |
| **Description** | Users and creators follow other creators and their followers' count is increased |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and the user wallet is connected |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Go to NFT creator profile |  |
| **4** | Press follow button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | Followers count is increased on the database | Pass |
| **2** | The updated follower number is shown on GUI | Pass |

Table . Test Case- Unfollowing Creator

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User/Creator unfollow others | T7 |
| **Description** | Users and creators unfollow other creators and their followers' count is decreased |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and the user wallet is connected |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Go to NFT creator profile |  |
| **4** | Press unfollow button |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | Followers count is decreased on the database | Pass |
| **2** | The updated follower number is shown on GUI | Pass |

Table . Test Case- NFT Upvoting

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User upvotes the NFTs | T7 |
| **Description** | Users upvoted the NFT and top NFTs are shown on the popular page. |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and the user wallet is connected |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Press the upvote button on the NFT card |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | The vote count is increased on the database | Pass |
| **2** | Updated upvotes of NFT are shown on GUI | Pass |
| **3** | NFT ranking is improved on the popular page | Pass |

Table . Test Case- NFT Downvoting

|  |  |  |
| --- | --- | --- |
| **Test Case Id** | User downvotes the NFTs | T7 |
| **Description** | Users downvotes the NFT and NFT ranking is decreased on the popular page |  |
| **Requirement** | Internet connection, Metamask Installed |  |
| **Initial Condition** | The application is online and the user wallet is connected |  |
| **Step** | **Procedural Steps** |  |
| **1** | Go to Application |  |
| **2** | Click connect wallet button |  |
| **3** | Press the downvote button on the NFT card |  |
| **Step** | **Task and Expected Result** | **Status** |
| **1** | The vote count is decreased in the database | Pass |
| **2** | Updated upvotes of NFT are shown on GUI | Pass |
| **3** | NFT ranking is decreased on the popular page | Pass |

**Chapter # 7**

# **Conclusion**

This project aims to make a marketplace where digital collectibles can be created and traded between users. This project mainly focuses on the gaming content creators' niche. Therefore, all the NFTs present on the platform are in video formats. Which NFT is good is decided by the community present on the platform by upvoting that NFT. It also depends on how good the content of NFT is. During the development of the project, a lot has been learned, from the financial concepts to the development of blockchain networks which gave future insights into how revolutionary this technology is and how it can be used in other use cases.

## 7.1. Future Improvements:

The first version of the NFT marketplace is the basic one that is demanded by everyone in the Web3.0 space. But this is not the stop of the journey. A lot still can be improved in this marketplace by adding more and more features. Due to the human resource and time constraints, more modules couldn’t be added, but in the future, we intend to add these features to our platform:

### 7.1.1. Integrated DEX:

Right now, we are listing our token on some decentralized exchange from which users can purchase our token. But in the future, the DEX itself will be integrated into the marketplace. The application will connect with our smart contracts instead of any outside smart contracts.

### 7.1.2. Staking Module:

There is a new concept of staking, where users can stake their NFTs and in turn get rewards in the form of tokens. During staking the NFTs are locked for a specific time. The more time NFT is locked more is the user rewarded for it. It also increases the value of NFT.

### 7.1.3. Interoperability:

Interoperability is one of the issues in existing blockchains. Blockchains are not connected. For this reason, we are not able to transfer our assets from one blockchain to the other one easily. There are ways like using bridges to transfer the assets but they cost too much in the gas fee. Right now, LayerZero networks have made their custom smart contracts where there is no need for a bridge and users can transfer their assets between EVM compatible blockchains. In the future, this module will also be included so that more users are comfortable using our platform on different networks.

# ***References:***

[1] OpenGeeksLab. An Ultimate Guide to NFT Marketplace Development (2021). Available at:

<https://opengeekslab.com/blog/nft-marketplace-development/> [Accessed 15 October 2021]

[2] Solidity Docs (2021). Available at <https://docs.soliditylang.org/en/v0.8.13/> [Accessed 1 December 2021]

[3] Node.js. Node.js. Foundation, N. (2018). Available at: <https://nodejs.org/en/> [Accessed 15 December. 2021].

[4] ANNA GOTSKIND. Decentralized Finance and the Power of Smart Contracts, 13 December 2021. Available at <https://researchblog.duke.edu/2021/12/13/decentralized-finance-and-the-power-of-smart-contracts/> [Accessed 16 December 2021]

[5] React.js Docs (2021). Available at <https://reactjs.org/docs/getting-started.html> [Accessed 24 December 2021]