**SYSTEM REQUIREMENTS**

1. **Functional Requirements**
   1. **User Registration and Authentication**

User should be able to see fields like username, email, password, confirm password on the sing-up page.

Users should be able to create an account using a valid email id or through MFA methods.

Confirming user’s email id by sending a verification link or sending an OTP to the mobile number to validate.

User should be able to securely login using email id and password.

User should be able to reset password in-case they have forgot it or wants to change

* 1. **NFT Creation and Minting**

Verified user should be able to create/mint NFTs.

User should be able to upload images, videos or other media files required for creating a NFT.

Efficient mechanism should be in place to prevent unauthorized minting.

User should be able to provide metadata for the NFTs during minting process

The system should be able to integrate with the appropriate blockchain network to perform minting process.

* 1. **NFT Listing and Buying**

Seller should be able to list down their NFT for selling.

Seller should be able to set a base price for the NFTs they are listing for sale.

Seller should be able to update their NFTs listings.

Buyer should be able to classify NFTs into different categories.

Buyer should be able to see the description of the NFT.

Buyer should be able to see transaction history that has been made.

The system should be able to suggest base price of the NFT to the owner while listing the NFT.

* 1. **Fractional Ownership**

The system should allow NFT owners to decide and create shares of the NFT.

User should be able to mark and create fractional shares to the NFT.

User should be able to list their fractional shares.

Fractional owners should be able to sell their shares and buy another fractional share.

Fractional owners should be able to see various parameters of the NFT like description, current value

* 1. **NFT Rentals**

NFT owner should be able to make rental listing for their NFTs.

Owner should be able to define time period and rental cost for their listed NFTs.

Owner should be able to demand security deposit or a collateral from the renter before renting to the renter.

Renter should be able to access NFT description and current price of the NFT during rental period.

Renter should be returned security deposit or a collateral as soon as rental time period gets over.

* 1. **Dynamic Pricing**

The system should be able to monitor market trends of the similar NFTs.

The system should be enabled with mechanism to set NFT prices based on the current market trends.

* 1. **NFT Insurance**

Owner should be able to explore different NFT recovery schemes based on the value, rarity and type of the NFT.

The system should be enabled with mechanism to suggest owner with best possible scheme available based on the features of the NFT.

User should be able to purchase insurance plan within the system, no third party should be involved.

Once payment is done, recovery scheme should be activated with the desired NFT.

The system should provide clear documentation of the insurance scheme with all terms and conditions.

Owner should have the option for renewal of the scheme once the scheme is expired

* 1. **NFT Staking**

NFT holders should be able to participate in staking pools where they can safe guard their NFTs.

NFT holders should be able to transfer ownership of their NFTs to the staking pool contract for the staking period.

The system should be able to clearly display potential rewards for different staking choices.

The system should be able to track participants in each staking pool and display their staked NFTs.

1. **Non – Functional Requirements**
   1. **Security**

The system should be able to store and transmit user data securely through encryption mechanisms.

The system should be able to ensure accountability and traceability.

The system should be able to provide access the user for the authorized data and functionality.

Implement a bug tracking system that allows users or administrators to report issues and resolve.

* 1. **Availability**

The system should be available and accessible for 24x7 for the user.

The system should be able to send an email regarding maintenance of the system well in advance.

The system should be designed in an efficient manner to minimize downtime and reach maximum availability.

* 1. **Performance**

The system should be able to provide response to the request made by user very quickly.

The system should be able to withstand with the increasing number of users.

The system should be able to handle sudden increase in user and request.

The system should be implemented with a mechanism to reduce load time for frequently accessed data.

* 1. **Reliability**

The system should be able to work even when there is any hardware or software failure.

The system should be able to ensure the accuracy and integrity of the token data and owner’s information.

The system should be enabled with mechanism to regularly back-up critical data including token details, owner’s information and transaction history.

The system should be designed with efficient load balancing mechanism.

* 1. **Usability**

The user interface should have a very clear layout that guides user throughout the system’s functionalities.

The user interface should be interactive and consistent.

The system should ensure that the user interface responds quickly and accurate to minimize load time.

The system should provide training material if the user has any confusion in interacting with the interface.

* 1. **Compatibility**

The system should work on various devices including desktop, laptops, smart phones having different Operating System.

The system should be able to run on different web browsers.

The system should be adaptable to different screen resolutions therefore making the system to work flawlessly.

The system should have a fallback mechanism so that the alternate solution can be activated to run the system without compromising any feature.

* 1. **Scalability**

The system should be designed to support horizontal as well as vertical scalability.

The system’s database should be designed to handle increasing data over time.

The system should have a mechanism in which the system automatically scale up or down according to the load condition.

* 1. **Integration**

Specify the APIs and interfaces that the NFT management system needs to interact with, both for inbound and outbound data exchange.

Jot down the external service or platform that the system needs such as payment gateway, blockchain networks.

Define how the data synchronization will happen between system and external services.

Ensure that system’s interface should be compatible with external services.

* 1. **Maintainability**

Design the system with modular components that can be updated or replaced independently without affecting the entire system.

Ensure that after any update in the system, new issues are not introduced.

The bug tracking system should be regularly maintained.

Implement monitoring and analyzing system to check the performance of the system including user interface.