Design Doc V1

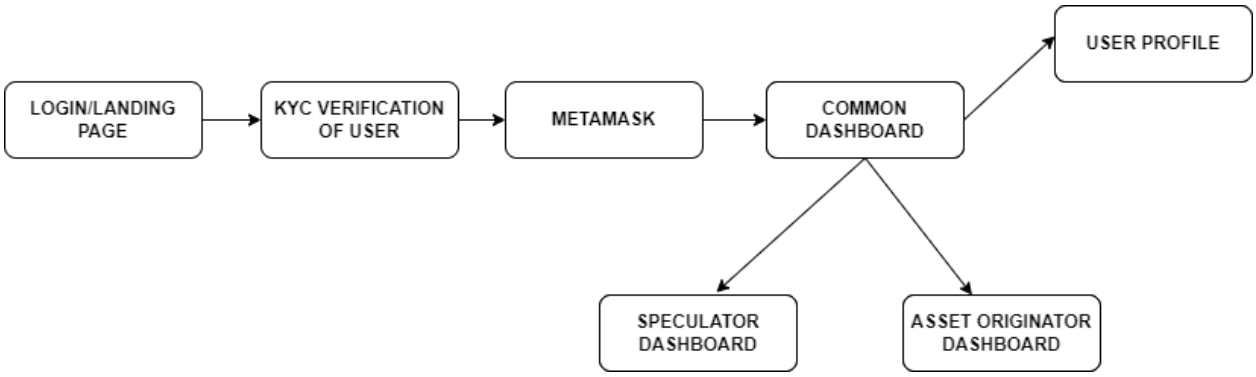
Own Your Home

Team 5

Naval Surange | Siddharth Jain | Akash C.R. | Divyansh Tiwari

Design Overview

Architectural design



We have a modular architectural design with six modules. The modules being Home Page(Login + Landing), User Profile, Common Dashboard, KYC page , Speculator Dashboard, Asset Originator Dashboard. Whenever a user visits our website they will land on HomePage and can access other modules through the navigation bar which connects them.The overall web app will be divided into various components containing APIs. By keeping the design modular it is helpful for any user to get the appropriate data required from the respective page easily and it also provides proper separation to show information from various databases in application. The system responsibilities are divided on the basis of the task they are serving and the definite category they fall into.

Responsibilities assigned to different subsystems:

1. **Homepage(Login + Landing page):**
   Any user who intends to use our webapp is asked to login using their metamask wallet. As soon as the credentials are entered, their metamask wallet is verified and are redirected to the Common Dashboard.

2. **KYC page:**
   All the Speculators and Asset Originators are required to verify their identities using KYC documents in order to make any transaction through our webapp. KYC verification is done using Verification email and with the help of secondary KYC verification

3. **User Profile :**
   All the details of the User's wallet and other personal details can be viewed on this page. And it is updated whenever transactions are made or any personal details are updated.

4. **Common Dashboard :**
   Details about the DAO and other statistical datas are provided here. And the user gets to choose how to get on board. Users can choose whether he/she wants to be asset originator or Speculator and depending on that they are redirected to corresponding pages.

5. **Asset Originator Dashboard:**
   This page is for users who are Asset originators. All the details of the assets of that particular user whose ownerships are traded as NFTs can be seen here. And the value of those are updated as and when required.

6. **Speculator Dashboard:**
   This page is for users who are Speculators and want to invest in asset ownership NFTs. In this dashboard all the details related to that particular user's purchased NFTs are shown and also their values are shown. It keeps getting updated whenever the value of property changes.

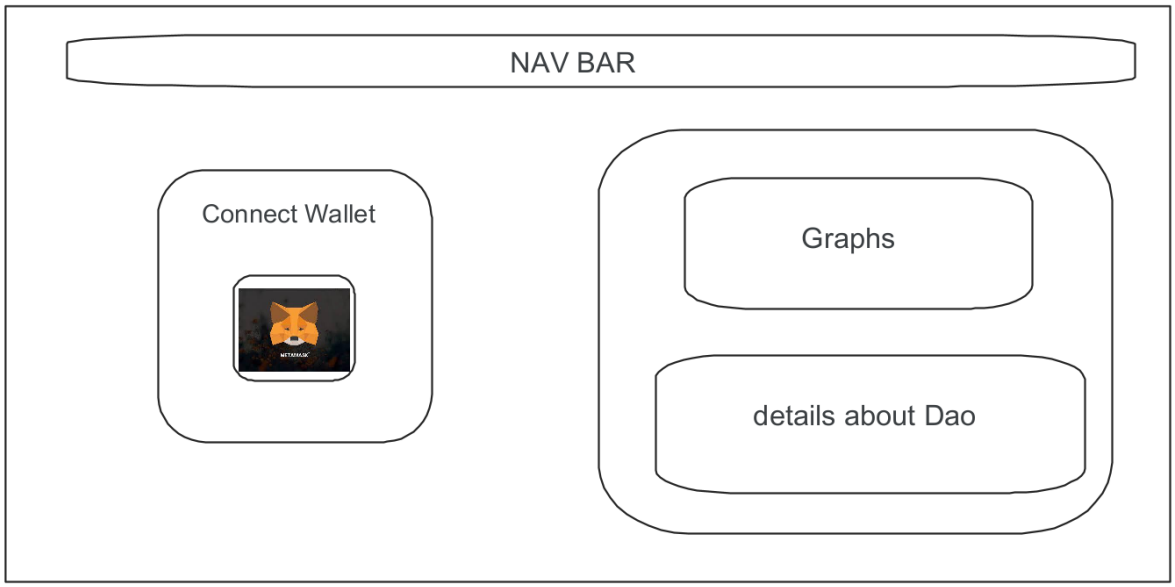
System interfaces

User Interface

The user will have the following webpage displayed as they open the website.

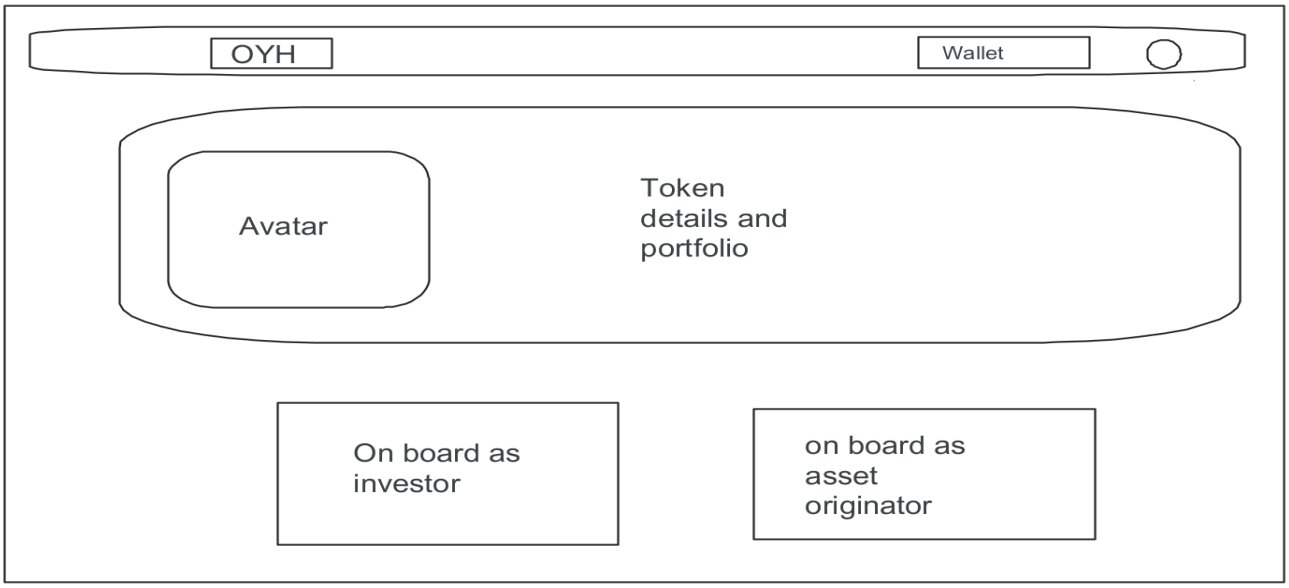
- The webpage will have the navigation bar with buttons serving the functionalities which are yet to be finalized.
- There will be a button to connect to the Metamask Wallet. Connecting to the Metamask Wallet is a Web 3.0 counterpart of login functionality.
- If a user has a metamask account, they will be logged into the website. Else, they will be redirected to the metamask website for a new account registeration.
- There will be grids sharing details about the DAO should the user like to see them.
- There will be graphs for a variety of analysis like token worth analysis.

login prompt



After logging in (that is connecting to the metamask wallet), the user will be directed to the following page where he gets to choose the purpose of his visit at the website.

- There will be a navigation bar having various buttons serving the purposes such as redirecting to the home page, wallet button having access to the wallet details such as amount of tokens owned, and an option to logout of the website.
- There will be a grid having an avatar. Users will get to choose their avatar for a better personalized experience on the website.
- The token details and the portfolio will be shared on a grid.
- The user will get to choose between being on board as an asset originator or an investor which will load the designated web pages about the choice.



The Investor Webpage will include:

- The details of assets on the chain in the form of NFTs.
- The price of NFTs and the share of NFTs available for sale.
- The details of properties of the concerned NFT including the address, the valuation, etc.
- The property rights obtained on
- Option to buy the NFT or a fraction of it in exchange for the tokens.

The Asset Originator Webpage will include:

- Steps to list a property or an asset on the chain.
- Button for KYC verification initiation and to check the verification status.
- Option to add RTL for each on chain property.

APIs

We will be creating primarily 3 API endpoints for our front-end to communicate with the backend server and the database.

a) **User Details API** - This API will be used to get the user details such as current KYC status, owned assets and the NFTs minted by the user.
   List of functions available -
   i) **GetUserDetails()** - returns the user details like userID, name, surname, Coinbase Wallet Balance, Profile url etc.
   ii) **GetAssets()** - returns assets owned by the user with given ID along with property rights and .

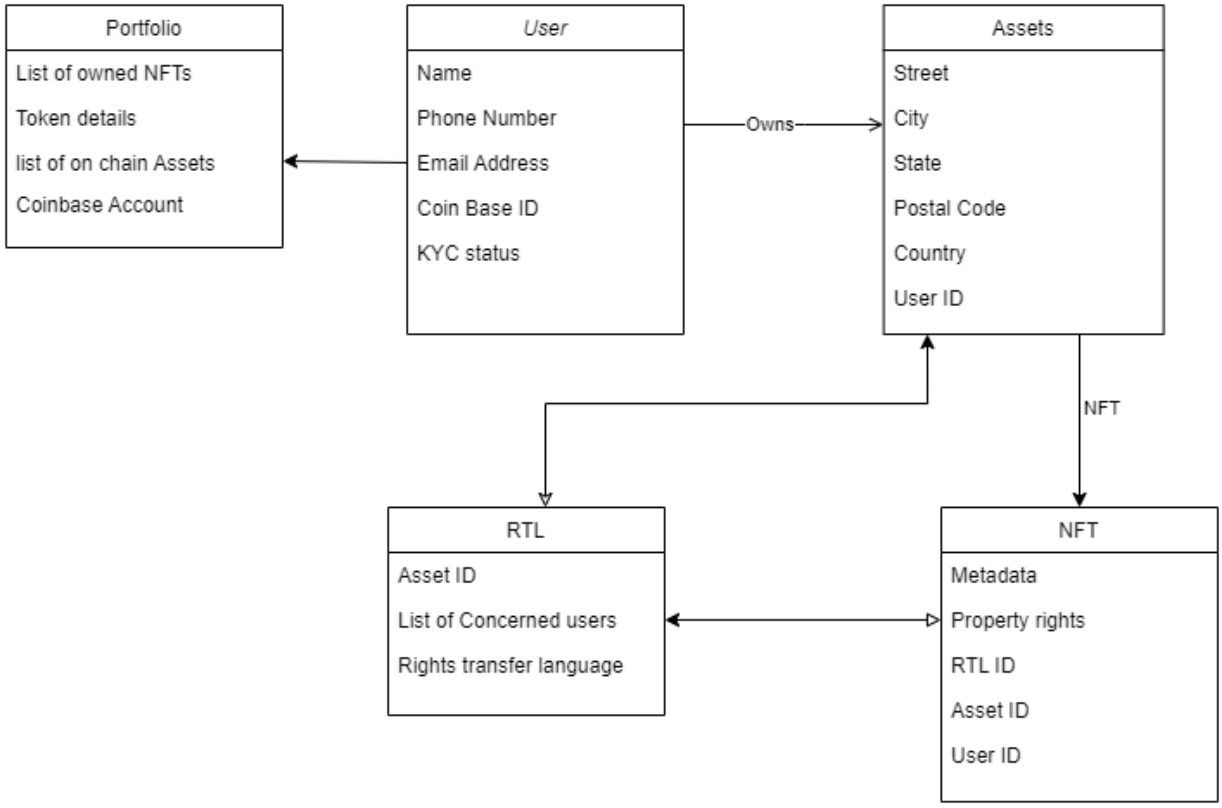
iii) **GetNFTs()** - returns NFTs owned by the user with a given ID along with its META-DATA.

b) **Asset Transfer API** - This API will be used to make possible transactions of buying or selling of assets on the blockchain.
List of functions available -
   i) **Transfer()** - the authenticated user can transfer funds,NFTs and assets from their wallet to the wallet of other users.
   ii) **GetTxnHistory()** - returns the transition history of the user with a given ID.

c) **NFT minting API** - API will be used to mint new NFTs for a given asset ID along with the metadata.
List of functions available -
   i) **Mint()** - mints and brings a new NFT for a given asset if and only if the asset was properly verified and got consent from each of the concerned persons.
   ii) **GetNFT() -** returns NFT details with a given Asset ID if it exists, else returns null if called by the user with proper privileges.

We are also using 2 API's provided by Coinbase and Metamask for our user authentication, validation, and KYC purposes.

Model

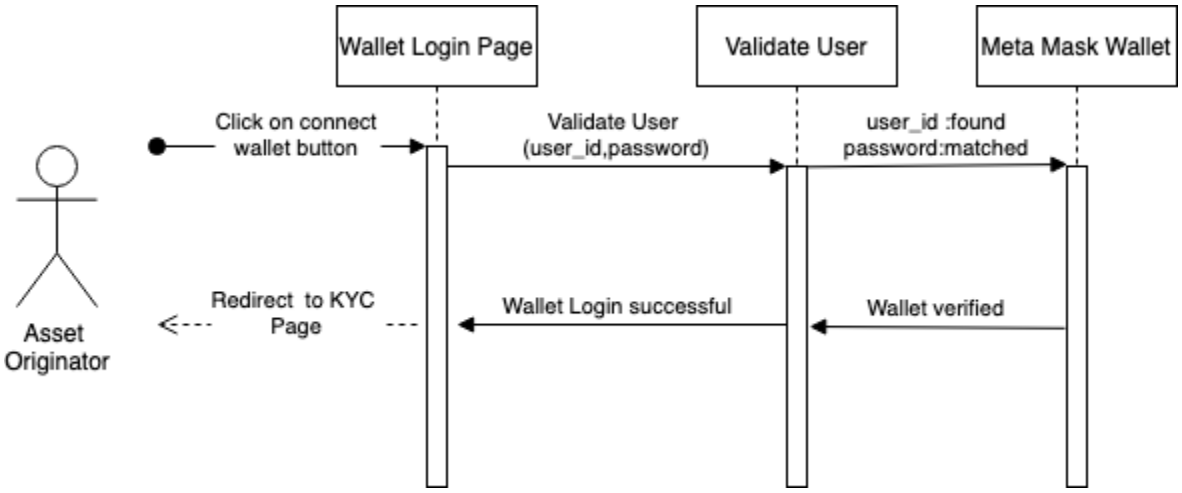


User	Class state ● Name ● Phone number ● Email Address ● Coinbase ID ● KYC status Class behavior ● Login() : For authorization and Login of user ● ConnectCoinbase(): connects with the coinbase account via Oauth. ● GetDetails(): returns the details of the current logged in user.
Assets	Class state ● Street ● City ● State ● Postal Code ● Country ● userID Class behavior ● QueueAsset(): For adding a new asset to the verification and minting queue. ● Validate(): For validation and consent taking of an asset in queue. ● Mint(): For minting of the validated asset with proper RTL. ● GetAssets(): For getting assets details like address,
Portfolio	Class state ● List of owned NFTs ● Token details ● List of on chain Assets ● Coinbase Account Class behavior ● GetNFTs(): returns NFTs owned by a user with a given user ID if authenticated properly. ● GetTokens(): returns Token details and balance of given user ID if authenticated properly.

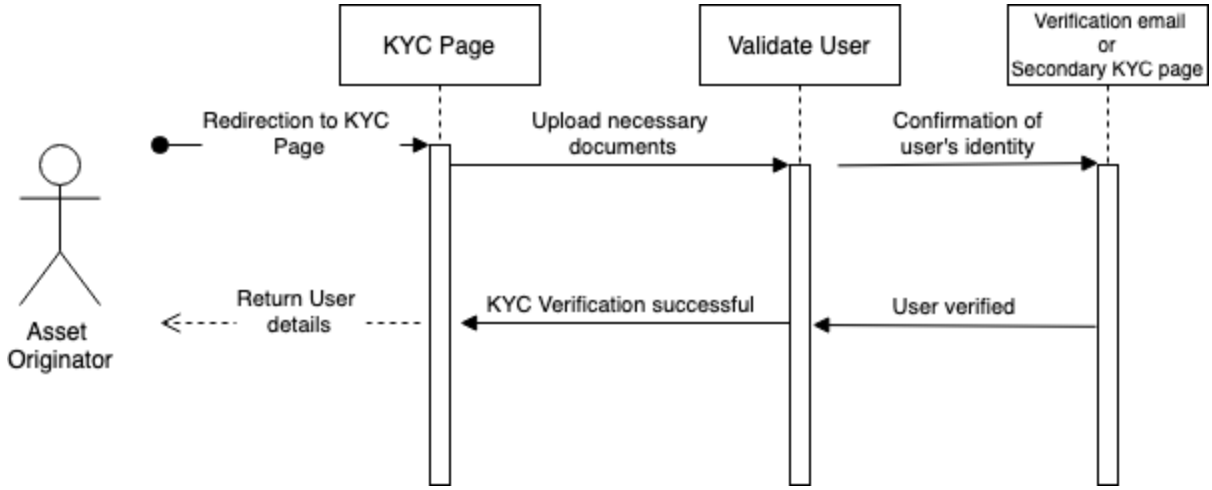
	● GetAssets(): returns Assets details of given user ID if authenticated properly. ● GetCoinbase(): returns Coinbase account details and balance of given user ID if authenticated properly.
RTL	Class state ● NFT ID ● Asset ID ● List of Concerned users ● Rights transfer language Class behavior ● CreateRTL(): Creates a new RTL(rights transfer language) for the NFT with a given ID. ● GetRTL(): returns the RTL for the NFT with a given ID if called by the user with proper rights.
NFT	Class state ● Metadata ● Property rights ● RTL ID ● Asset ID ● User ID Class behavior ● CreateNFT(): Creates an NFT for Asset with given ID and links it with given metadata, RTL, and User if asset and user both are validated properly. ● MintNFT(): mints the created NFT and brings it on ethernet mainnet. ● GetNFT(): returns NFT details with a given Asset ID if it exists, else returns null if called by the user with proper privileges.

Sequence Diagrams

Use Case Number:	UC 01
Use Case Name:	Connection to wallet
Overview:	The asset originator should be able to connect to the platform using the MetaMask wallet.
Actors:	Asset originators
Precondition:	First step
Post Condition:	The user should be logged in if authentication was successful or redirected to the metamask official webpage for help.

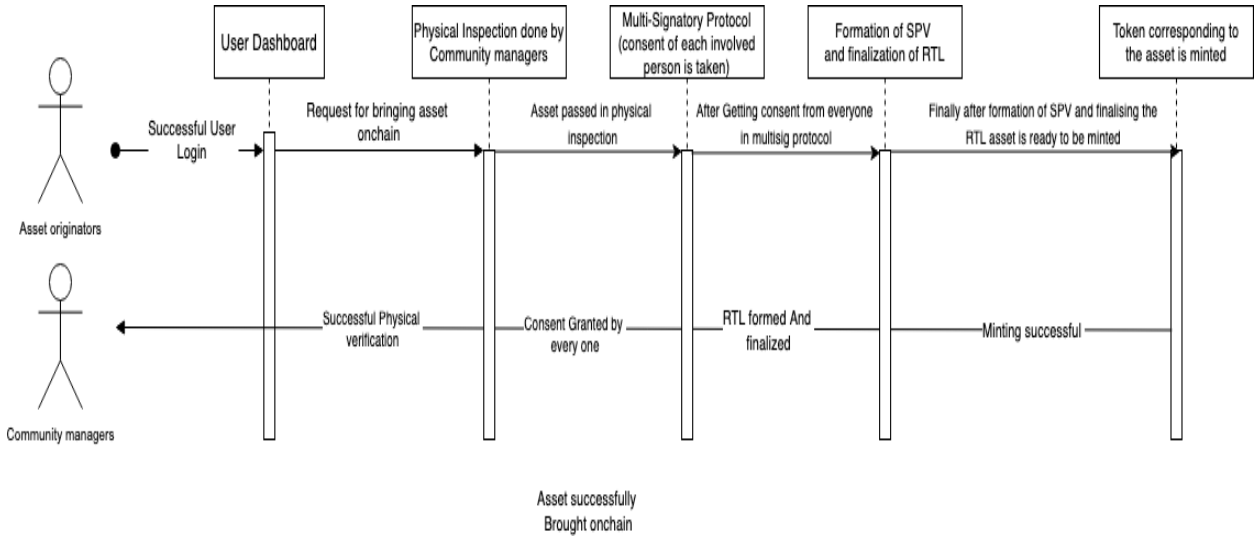


Use Case Number:	UC 02
Use Case Name:	Redirection to KYC page
Overview:	After connecting in, the Asset originator should be redirected to do a secondary KYC to confirm the person's identity.
Actors:	Asset originators
Precondition:	Users should have either registered freshly or signed up for KYC.
Post Condition:	Users will get a KYC verification email or will be redirected to the secondary service’s help page.



Use Case Number:	UC 03
Use Case Name:	Bringing the asset to the block chain
Overview:	The asset originator should get an option to bring their property on the blockchain
Actors:	Asset originators, community managers
Precondition:	KYC should be complete and user should be logged in
Post Condition:	A new property verification request should be placed after this.

Use Case Number:	UC 04
Use Case Name:	Physical verification
Overview:	The community manager should be able to see the assigned property verification requests placed by the asset originator and verify the properties by doing physical inspection.
Actors:	Community Managers
Precondition:	The Community Manager should have a database of property verification requests along with the supported documents.
Post Condition:	The Community Manager will pass the property verification onto the next stage.



Combined sequence diagram for UC-3 and UC-4

Design Rationale

- The dashboard design was changed to address simplicity. The user, after login, will get the option to proceed either as an asset originator or an investor. Thus resources will be loaded on the basis of the user's wish to act either as an asset originator or an investor which would make the website light.
- The KYC processes were shifted from requiring Government issued IDs and building a unique API to using Coinbase API.
    - The coinbase already has access to the users government IDs hence it is easier and more user friendly to simply just connect the coinbase account rather than reinventing the wheel.
    - This was done to satisfy the client’s needs.
    - This was done to address interoperability which happens to be an essential part in the Web 3.0 DAO system.
    - The project aims to build the webapp in a **no code/low code** fashion and choosing a pre-built API helped us to address it.
- Connections to the Metamask wallet and was done using the APIs provided by them.
    - Metamask is a popular wallet which has been in service for a long time now. It ensures safe transactions.
    - This API provides direct user authentication and validation.
    - Metamask is widely accepted by the users.
    - This is used to have our project to be in sync with the current Web 3.0 developments and address interoperability.