Pavan Kalam

16358242

COMP_SCI-5576-0001-44175-2024FS-Blockchain

Project:

TokenLand: A Blockchain--Based Real Estate Management system

Final Week Report – 7&8

Introduction:

TokenLand is a real estate marketplace that uses blockchain technology and Non-Fungible Tokens (NFTs) to perform property transactions. This decentralized application (DApp) allows users to tokenize real estate properties, list them for sale, and facilitate secure, transparent purchases.

Every property is a unique NFT token, which ensures immutable ownership records and streamlined transactions, by reducing intermediaries, enhancing transparency, and providing a global marketplace for property trading, potentially transforming the real estate industry.

This DApp is using Solidity for smart contract development, ethers.js for blockchain interactions, and a JavaScript frontend for user interface. It's currently deployed on a local Hardhat network for development and testing purposes.

Key Features:

- Properties are represented as unique NFTs on the blockchain, ensuring authenticity and immutability of ownership records.
- This platform supports two types of users - administrators who are the sellers, and they can mint new property tokens, and buyers who can only purchase listed properties.
- Users can connect their wallets (i.e., MetaMask) to interact with the smart contract and perform transactions.
- Property owners can list their tokenized properties (i.e., minted properties) for sale, while buyers can browse and purchase available properties.
- This platform maintains a temporary record of all property-related transactions.

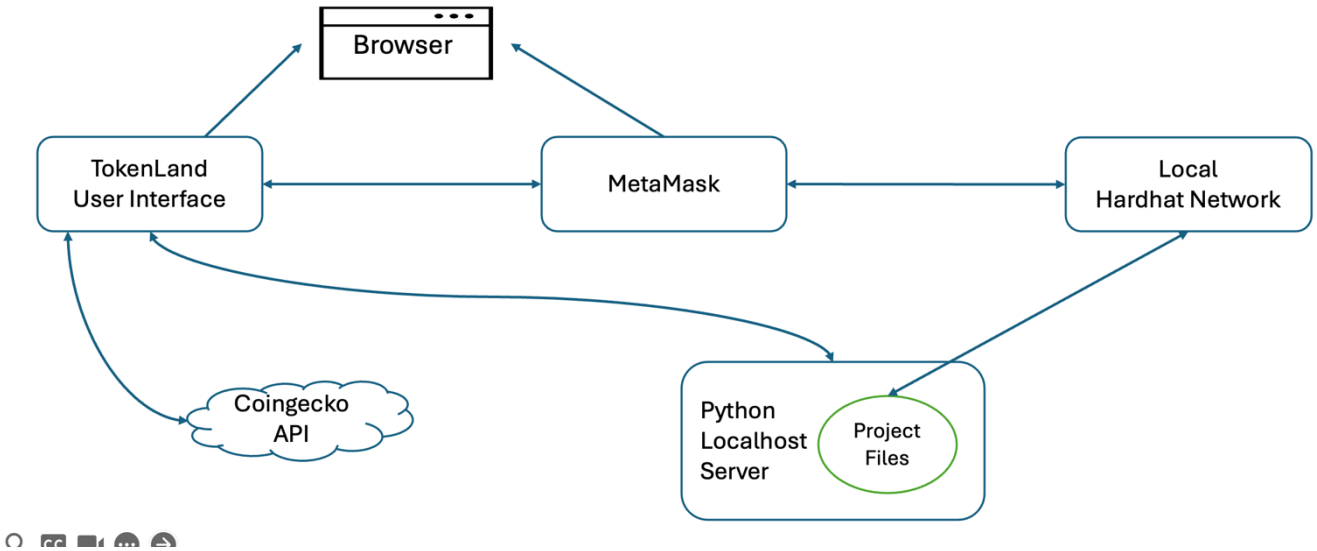
System Manual:

→ System Architecture:

- The smart contract written in Solidity code and saved under ‘contracts’ directory.
- It inherits from ERC721 (for NFT functionality) and Ownable (for access control).
- Key functions of this smart contract is as below,
  - mintProperty: To create a new property NFT.
  - buyProperty: It allows users to purchase a property.
  - listPropertyForSale: Lets owners list their property for sale.
  - getProperty: To retrieve property details.
- Here Frontend was developed using React.js. Also, frontend provides a user interface to interact with smart contract.

- Also, hardhat network has been used to create a local blockchain network.
- The TokenLand smart contract is initialized using the contract address, ABI, and signer.

SYSTEM ARCHITECTURE



→ **Wallet Integration:**

- MetaMask wallet has integrated with the blockchain, to enable secure interactions. This integration is crucial for authenticating users, signing transactions, and managing digital assets (NFTs) within the platform.
- Wallet connection is initiated through the connectWallet() function.
- This function first checks the availability of MetaMask in the browser.
- Once the MetaMask is detected, this function requests access to the user's accounts, which can be available in hardhat node server.

- Then MetaMask triggers a popup, asking the user for permission to connect their wallet to TokenLand.
- After user approval, Web3 will initialize and acts as a bridge between Frontend, wallet and the hardhat network.
- Once all this, the connected wallet address, current block number and user balance will be fetched and shown in the interface.

→ **Local Deployment and Instructions:**

- First install the hardhat in local working directory using command prompt.
- Make sure not to install the sample project dependencies while installing the hardhat, it will cause trouble to project dependencies version support.
- Below command will install the supported versions of the project dependencies.
  npm install --save-dev "hardhat@^2.12.7" "@nomiclabs/hardhat-ethers@^2.2.3" "ethers@^5.7.2" "@nomicfoundation/hardhat-toolbox@^2.0.0"
- Deployment script has been created in a separate folder called ‘scripts’.
- Now deploy the smart contract using the hardhat deploy command.
- Make sure run the hardhat node command to start the hardhat server in another command prompt, before deploying the smart contract.
- Finally, update the ‘contract address’ and ‘abi’ in ‘app.js’ code.

→ **Security considerations:**

- Here we had created a user authentication page for buyer and seller.

- Also, contract verification has implemented to compare the deployed contract address and the wallet address.
- If this is not matched user can’t get access to buy the properties.

→ **Transaction Confirmation:**

- All transactions require confirmation in the user's wallet, providing an additional layer of security.

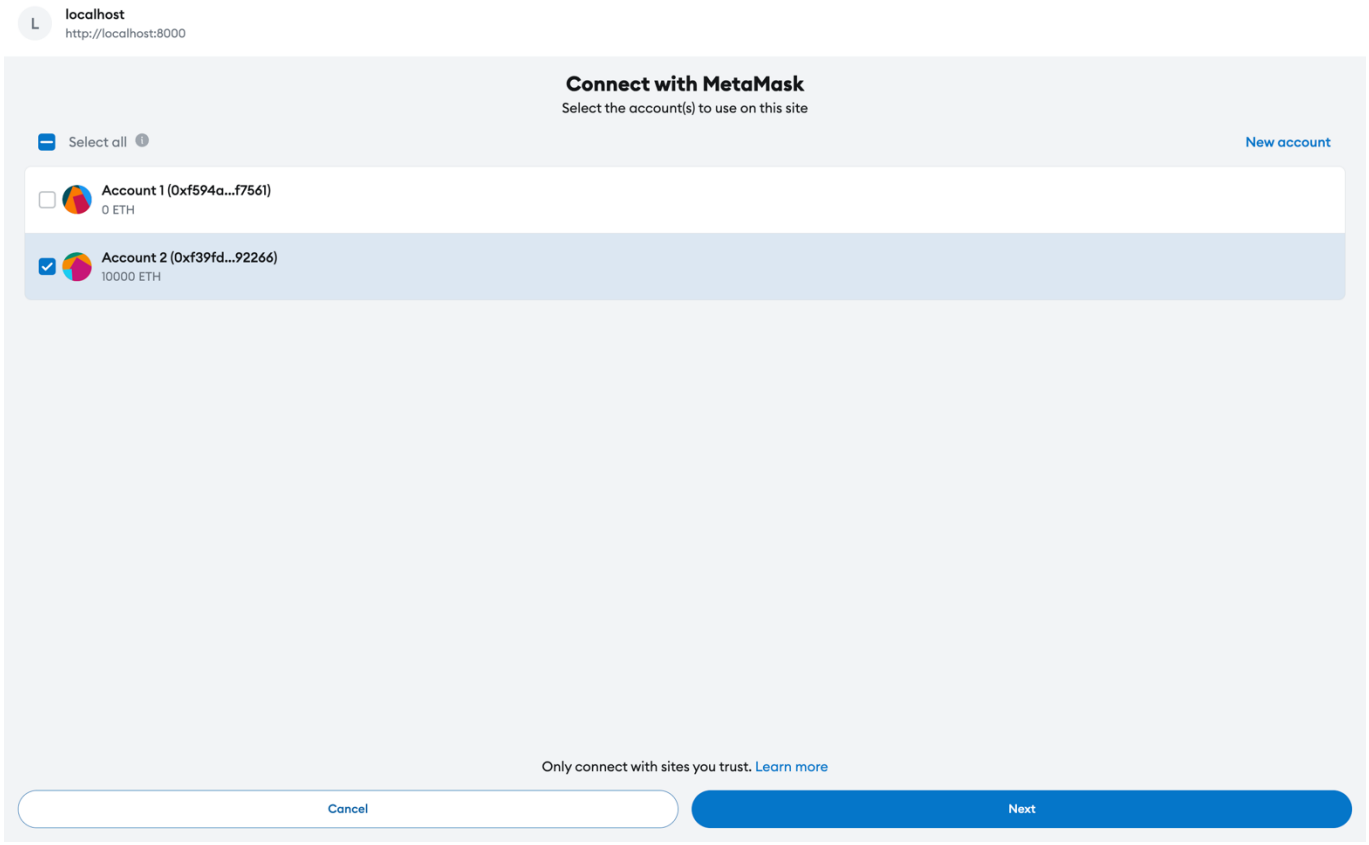
User Manual:

→ **Installing MetaMask:**

- First install the MetaMask wallet extension in the browser.
- Add a new network in the MetaMask to interact with hardhat with hardhat network configurations.

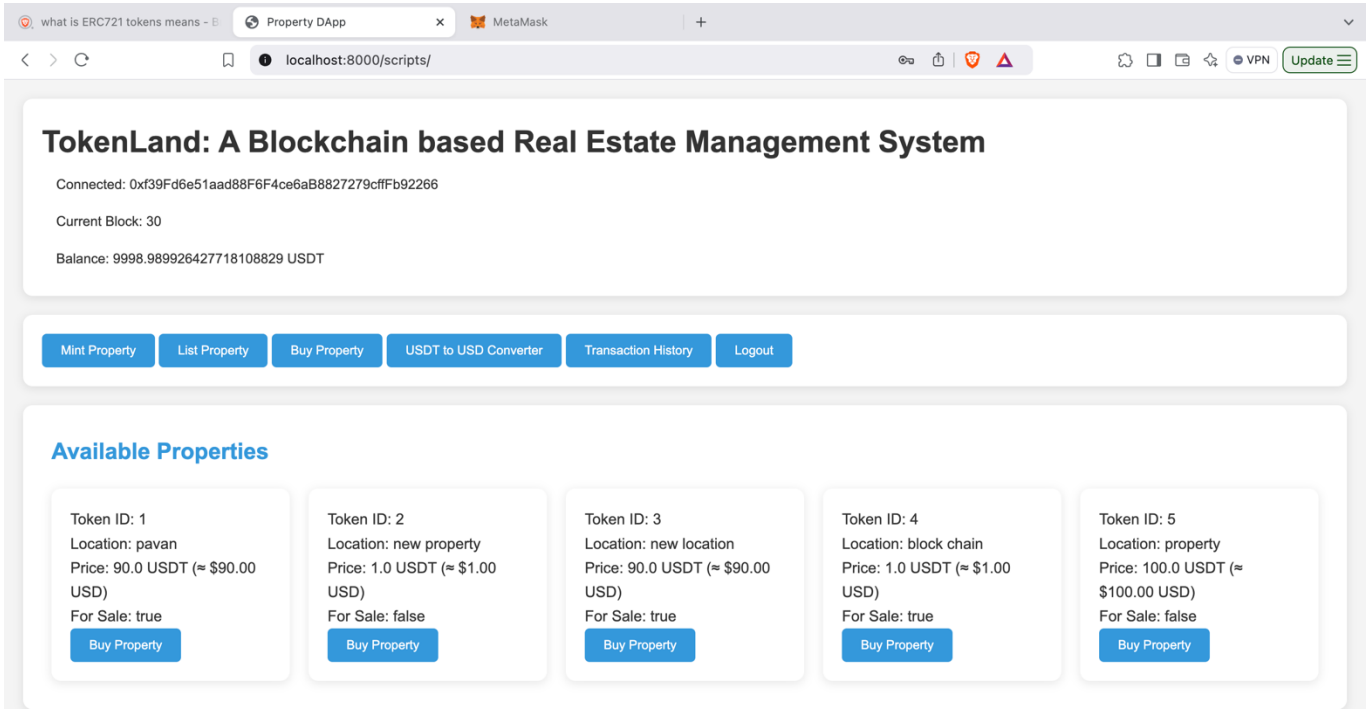
→ **Connecting Wallet to DApp:**

- Import the account to MetaMask through one of the private keys of hardhat network accounts, which can be available in the terminal where hardhat node is running.
- Make sure to connect to the deployed contract account.
- Once connected it will show the account balance and wallet address.



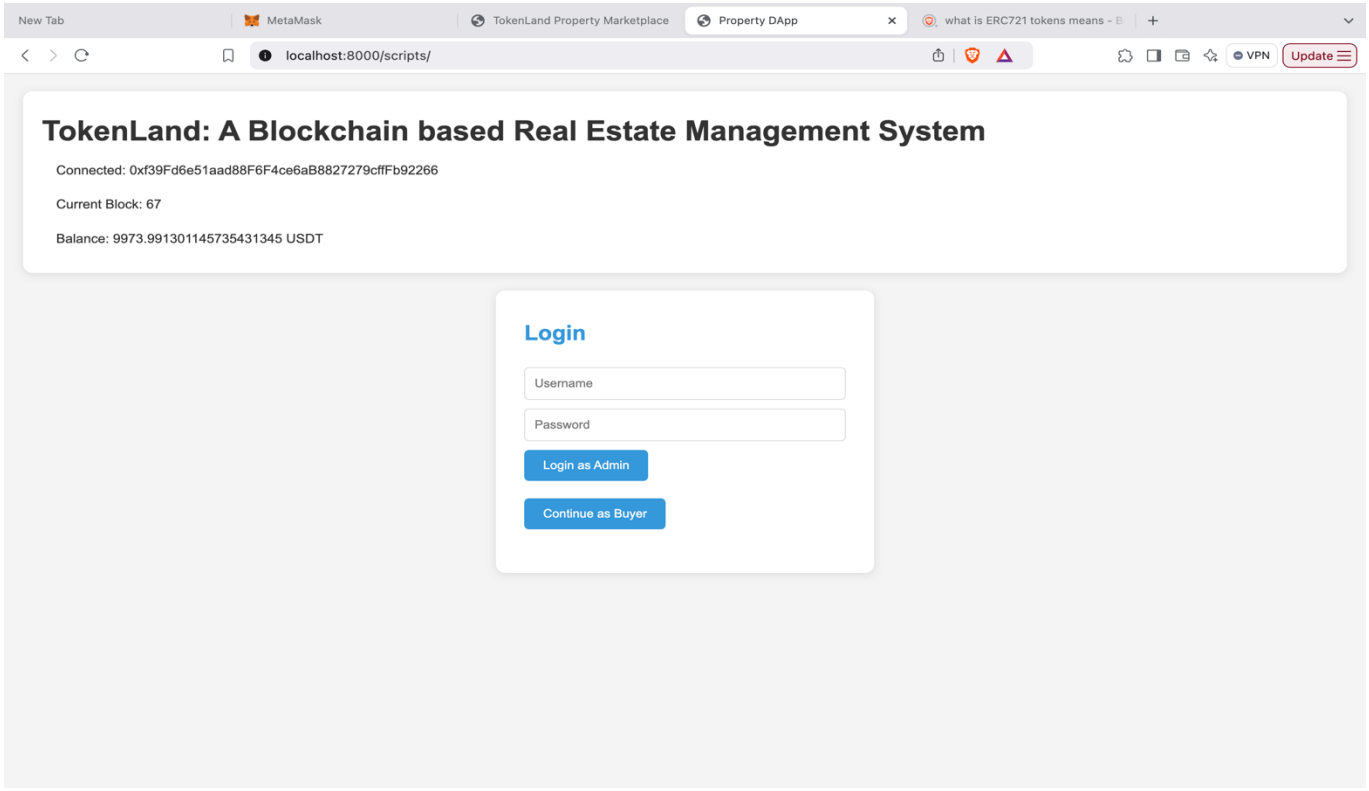
→ Interacting with User Interface:

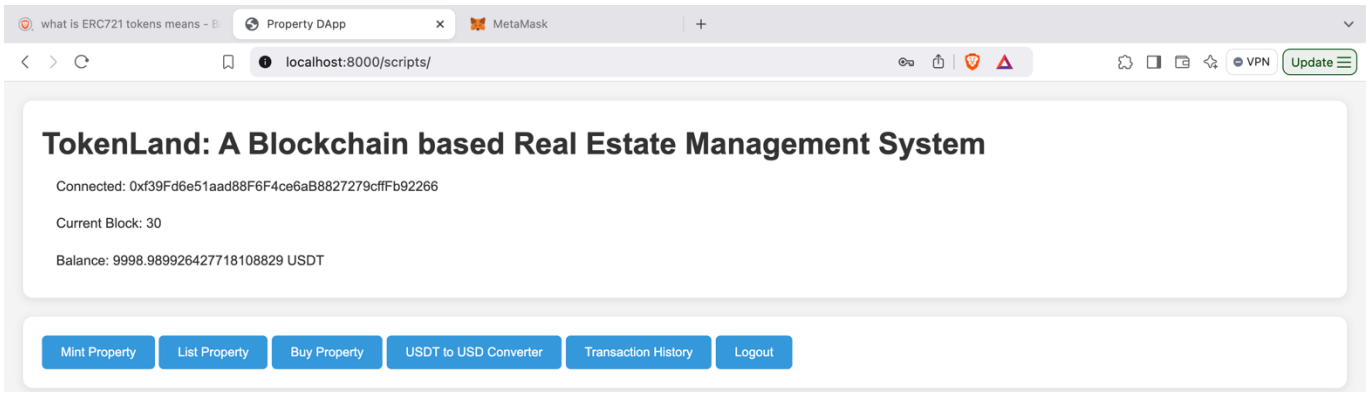
- Firstly, create and start a local server to access the user interface code.
- Here in this case python server has been used to access the local files in the browser.
- Once, the files are loaded in the browser the interface will be as below,



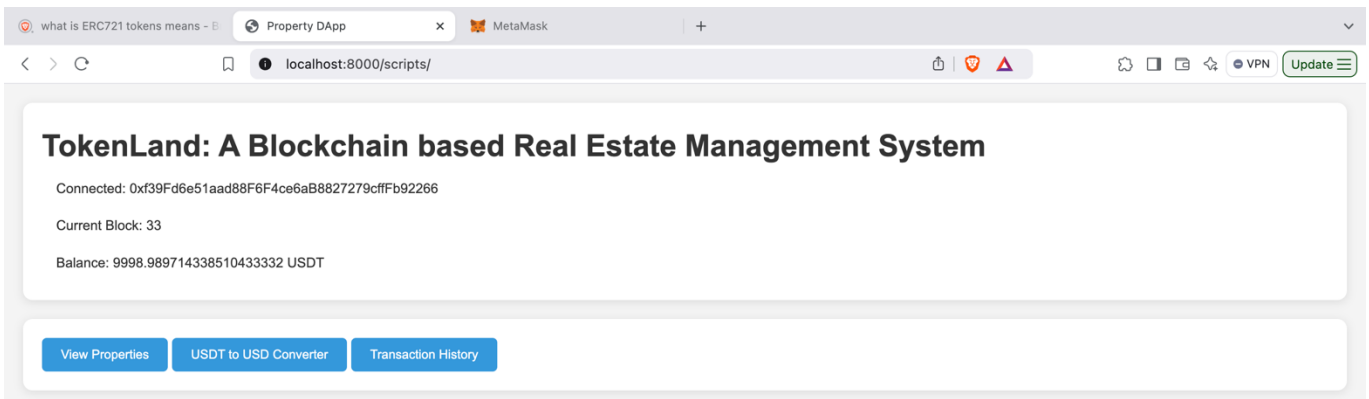
→ Admin and Buyer roles:

- Once, interface was loaded in the browser, user must connect to the wallet.
- After connection, the interface will load authentication page, for the user to login either as ‘admin’ which is seller or as only a ‘buyer’.
- To login as admin user must provide the admin credentials, to get complete options for minting, listing, buying and everything as shown below,





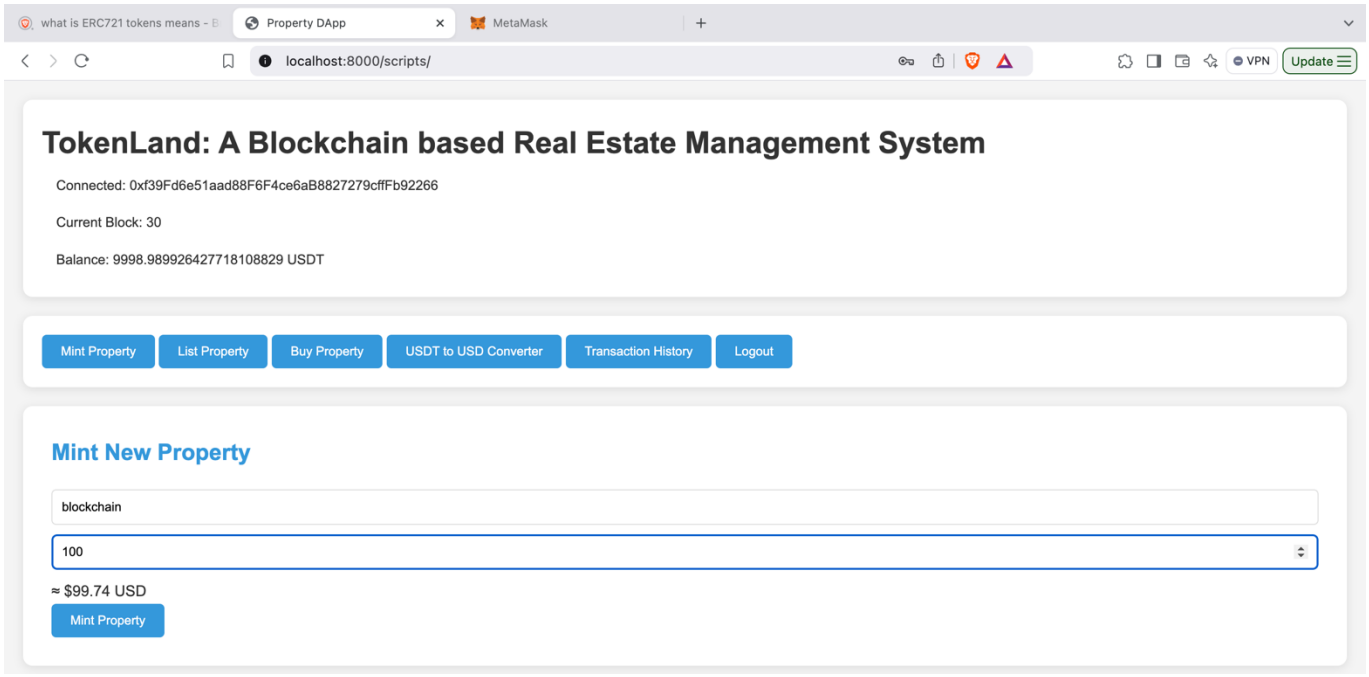
- If the user wants to be only a buyer, then the user can directly click on the ‘Continue as Buyer’ button in the user authentication page.
- In this case the user will only have access to buy the property, check the conversion rate and the transaction history he made.
- The buyer option will be as below,



→ Features of Smart Contract interactions with the interface:

1. Minting Properties:

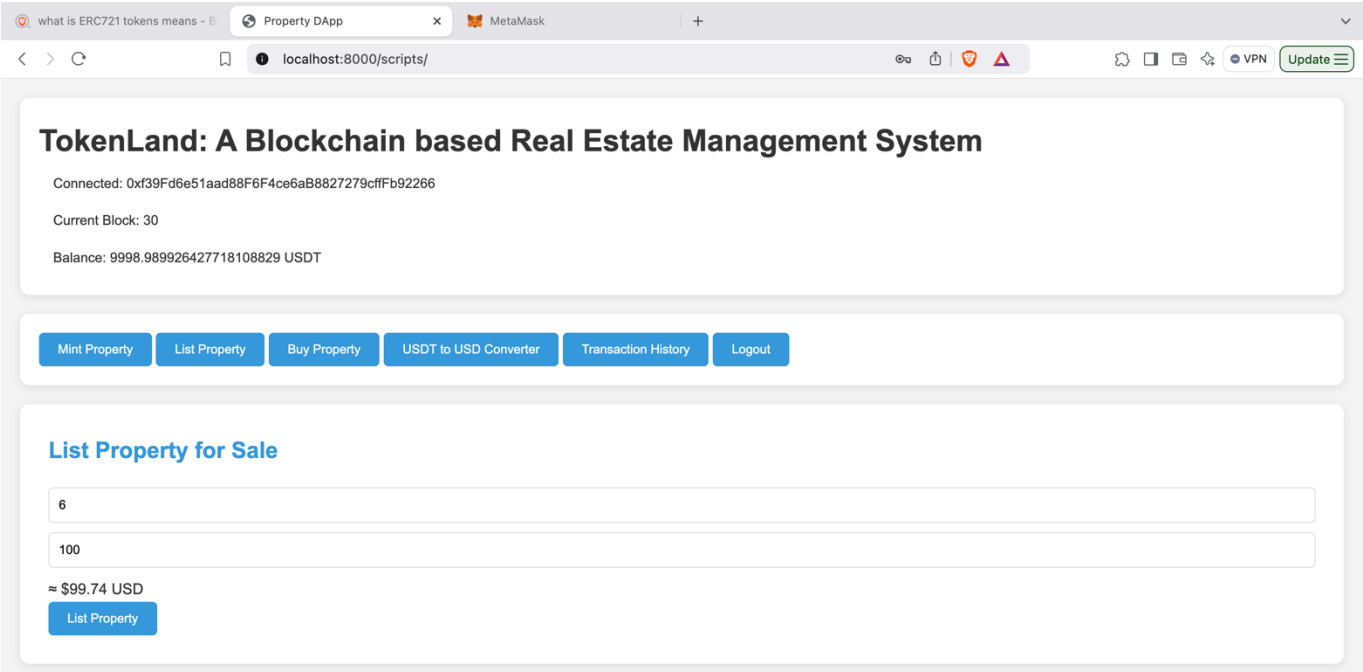
- Only property sellers have the access to mint the property.
- During minting provide the property location and value of the property.
- Once, details are provided click on ‘Mint Property’ button.
- MetaMask will pop up for confirmation, here click on confirm.
- Then the property will be minted successfully.
- Once, it is successful it will generate the token Id for the property.



2. Listing Properties:

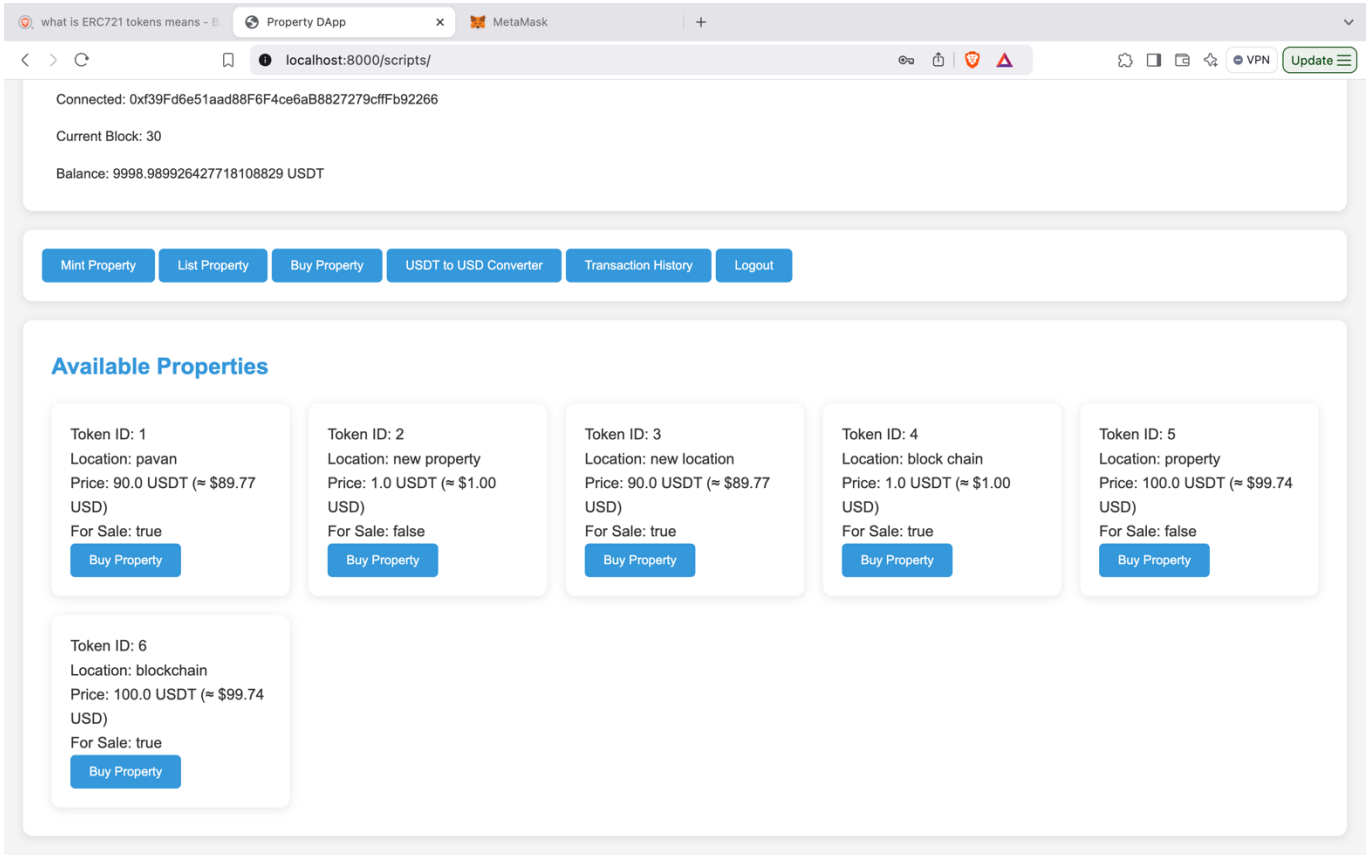
- After minting the smart contract will generate a NFC token and this token has a unique token id.
- Using this token id the seller can list the value for the minted property.
- Also, the user can only list the property with the value less than or equal to the minted property value, and not more than that value.

- After entering the property listing details, click on ‘List Property’ button.
- Again, MetaMask will pop up for transaction confirmation.



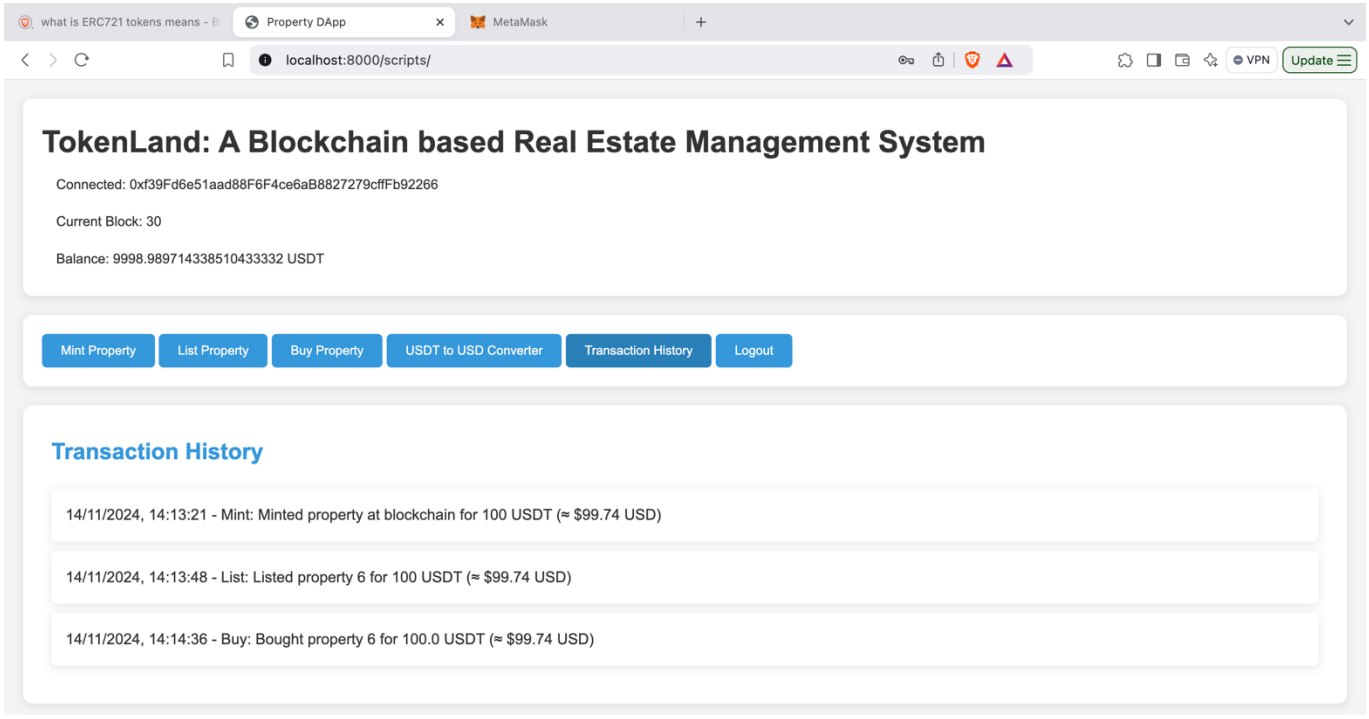
3. **Buying Properties:**

- Buying the property can be available to any user which is the user be either admin or buyer.
- In this step the property is ready for selling, with the property information like value of the property, availability.
- Now any user can buy the property by just clicking on the ‘Buy Property’ button.
- If anyone already purchased the property, it would show property availability as ‘false’, if the property is available, it will show the availability as ‘true’.
- Even for buying the property MetaMask will pop up for transaction confirmation.



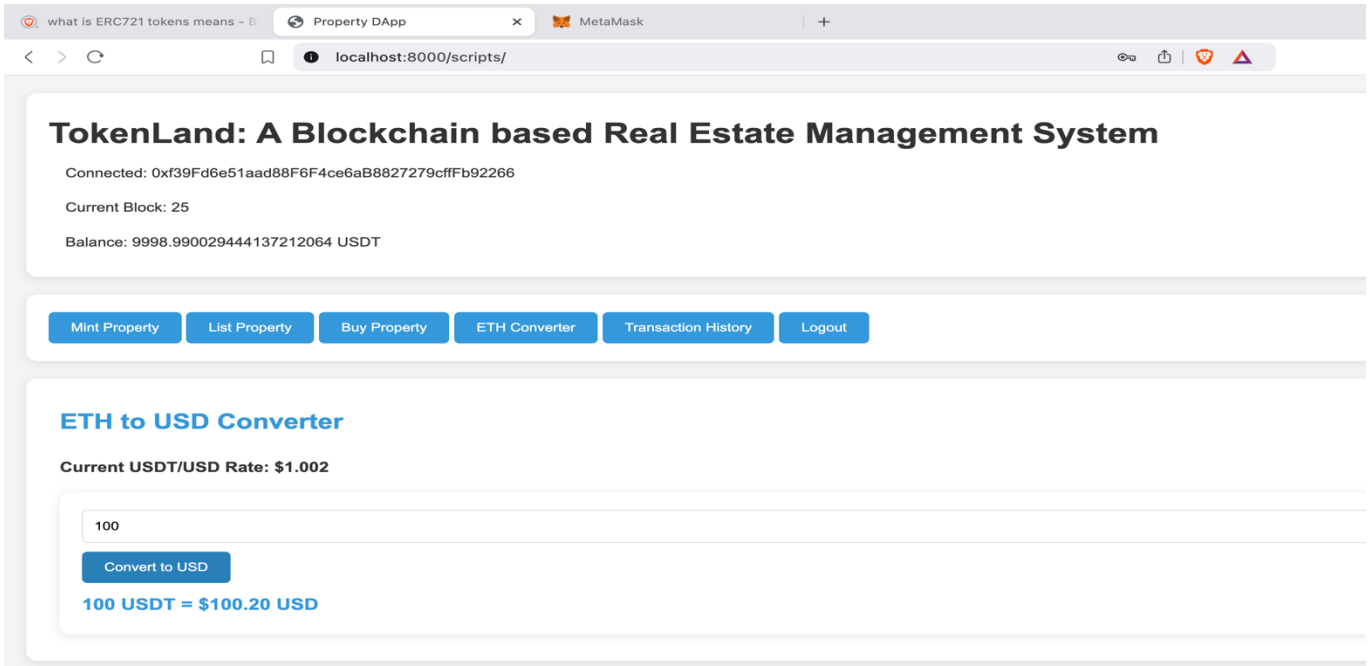
4. **<u>Transaction History:</u>**

- As we are using local hardhat network, it can't be possible to track all the transactions.
- So, Transaction History option has been introduced to track the minting, listing and buying information.
- For secure tracking, use MetaMask to check more informative details about the transaction.



5. **USDT to USD converter:**

- In this project we just implemented a USDT currency for property transactions.
- To make the user to understand the current USD rate, this option will always be available any user.

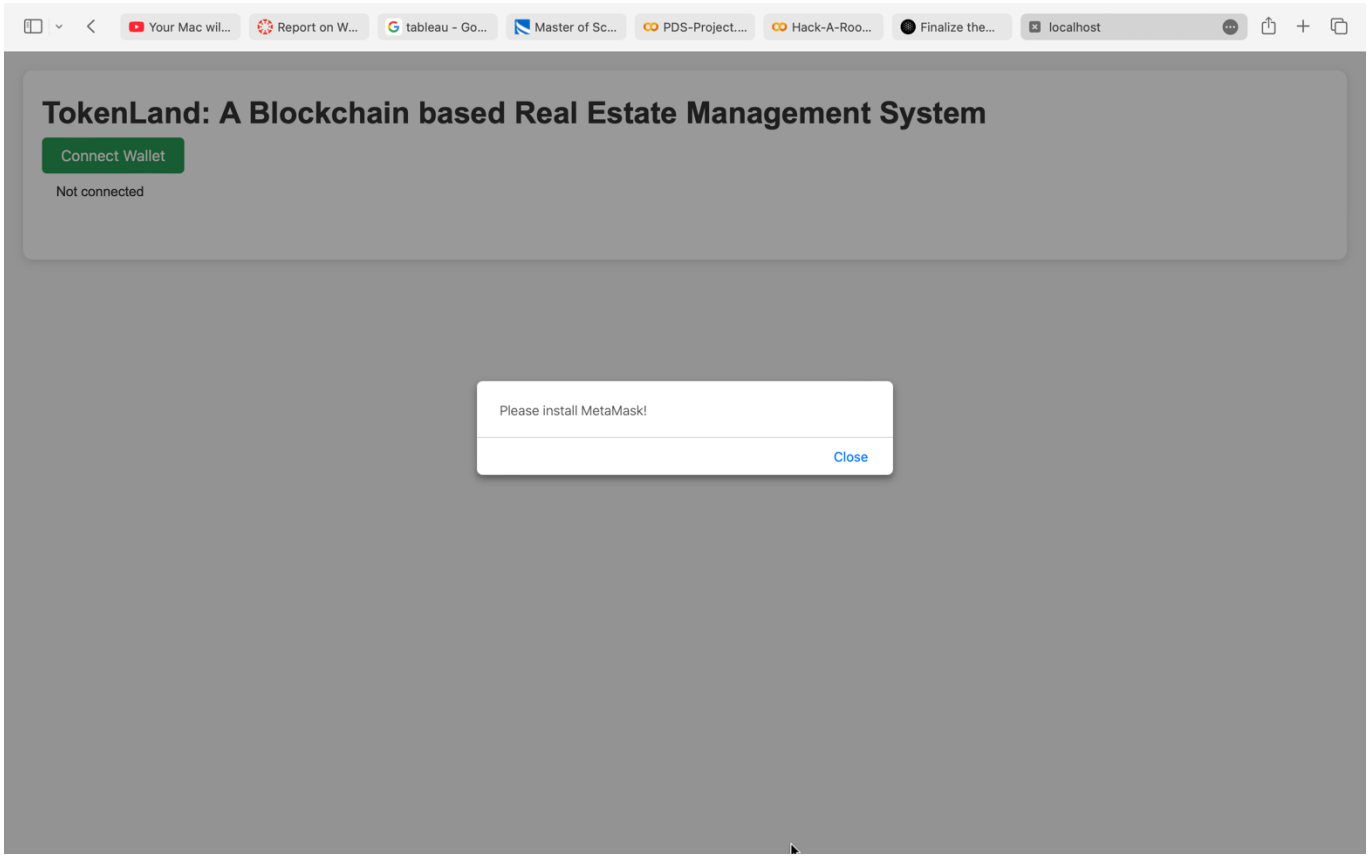


API Used for Currency Exchange:

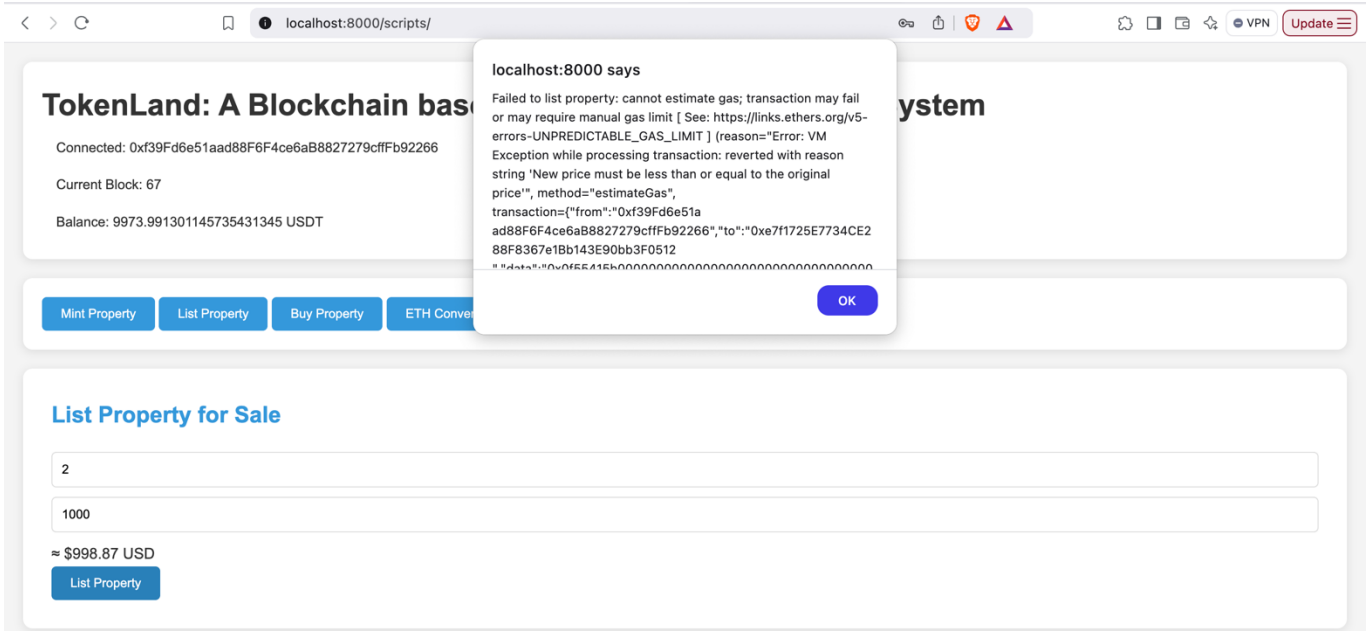
- The property exchanges will be performed through using of USDT currency.
- To make this user friendly, currency exchange API has be included in the project for providing the live exchange value updates.
- Here in this case ‘coingecko’ API has been used for live exchange updates.
- The exchange rates has been included in every option for immediate USD value identification to the users.
- The exchange currency rates can be observed in minting, listing, buying and all the options.

Error Handling:

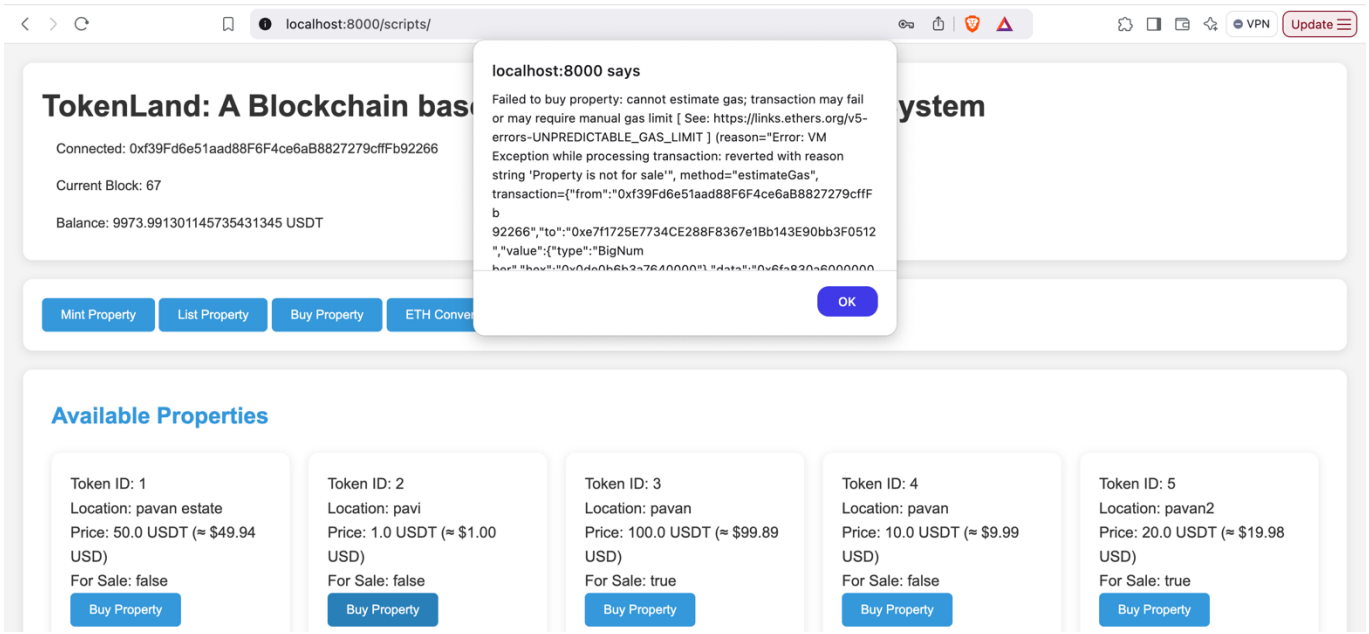
Case-1: If the MetaMask is available in the user browser, the error will be as below,



Case-2: If the user tries to list the property with more value than the minted value of the property, then the error will be as below,



Case-3: If the user tries to buy the property which is not available, that is the property is already sold, then the error will be as below,



Conclusion:

TokenLand project was successfully implemented, by property tokenization as NFT, and the potential usage of blockchain technology in this project makes real estate transactions can be implemented without the intermediates, also to maintain the transparency and reducing the potential fraud. This also enables fractional ownership of properties. The way we purchase, sell, and manage real estate assets around the world can be able to change by TokenLand.

GitHub:

Final presentation and the all the deliverables of the project is uploaded to GitHub, and the follow the link below,

https://github.com/pavan-kalam/Blockchain-based-realestate-management-system