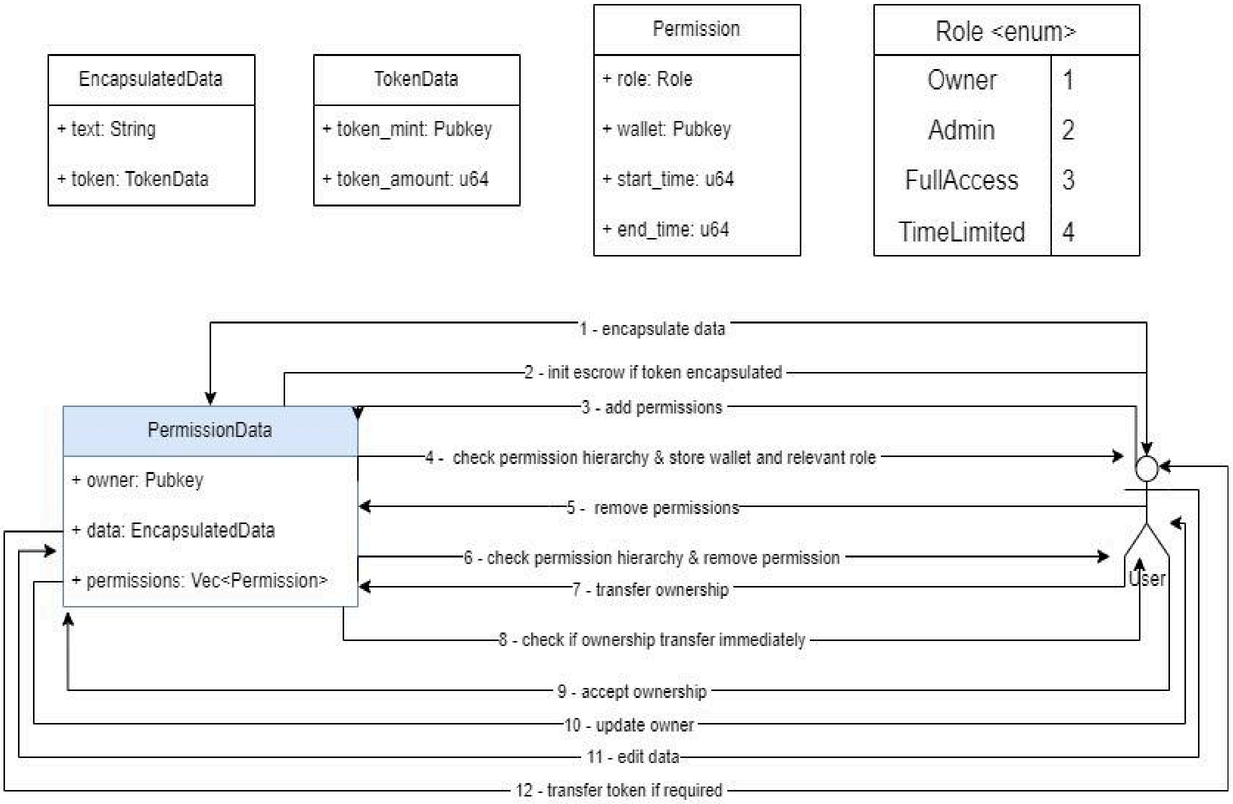
Protocol POC Requirements

- The protocol shall allow owner to encapsulate data
- The protocol shall allow the owner and admin(s) to add more wallets that can access the encapsulated data
- The protocol shall allow the owner and admin(s) to set a permission role for the new wallets. Different wallets can have different roles for the same encapsulated data.
- The protocol shall allow the owner and admins(s) to remove permission of a wallet from accessing encapsulated data.
- The protocol shall allow the owner to transfer ownership of encapsulated data to a different wallet.
- The protocol can let the owner delegate ownership of encapsulated data to a different wallet with a delegated ownership time in the future.
- The protocol can allow the delegated owner to accept ownership when the delegated time is reached.
- The protocol shall let wallets with access to the encapsulated data to update the data.

Overview



- 1 - encapsulate data:
    - Owner encapsulates the data
    - Data can be text, fungible and non fungible SPL tokens.

- 2 - init escrow if token encapsulated:
    - If an SPL token is encapsulated, the platform initialises an escrow and deposits the token in a vault authorized by the escrow.
    - No escrow is needed if textual data is encapsulated.

- 3 - add permissions:
    - Only the owner or admin(s) can call the add permission function.
    - The wallet address and corresponding permission role should be passed as argument.
    - If the role is time limited access then the starting time and ending time of the access are also passed as UNIX timestamp arguments.

- 4 - check permission hierarchy to add:
    - The platform first checks if the caller has enough permission to call the function.
    - The permission checks if the input wallet already exists in the list of authorised wallets. If so, the wallet will check if the caller has enough permission to update the permission of the input wallet. If both passes, the platform updates the permission of the input wallet.
    - If the input wallet does not exist, the platform appends the input wallet in the list of authorised wallets along with its permission role.

- 5 - remove permissions:
    - Only the owner or admin(s) can call the add permission function.
    - The wallet address should be passed as an argument.

- 6 - check permission hierarchy to remove:
    - The platform first checks if the caller has enough permission to call the function.
    - The platform checks if the caller has enough permission to remove the permission of the input wallet. If so, the platform removes the input wallet from the list.

- 7 - transfer ownership:
    - Only the owner can call the transfer ownership function.
    - Owner needs to pass the new owner's wallet address as an argument.
    - Owner needs to pass the future ownership time if she wants to delegate ownership to a future time.

- 8 - Check ownership transfer:
    - The platform checks if the time argument is a future time.

    - If so, transfer ownership immediately.
    - If the input wallet exists then its role is updated and if it does not exist then it is added to the list as the new owner.
    - The role of the original owner is updated to the admin role.
  - If the time is a future time, the platform will save the ownership start time for future reference.

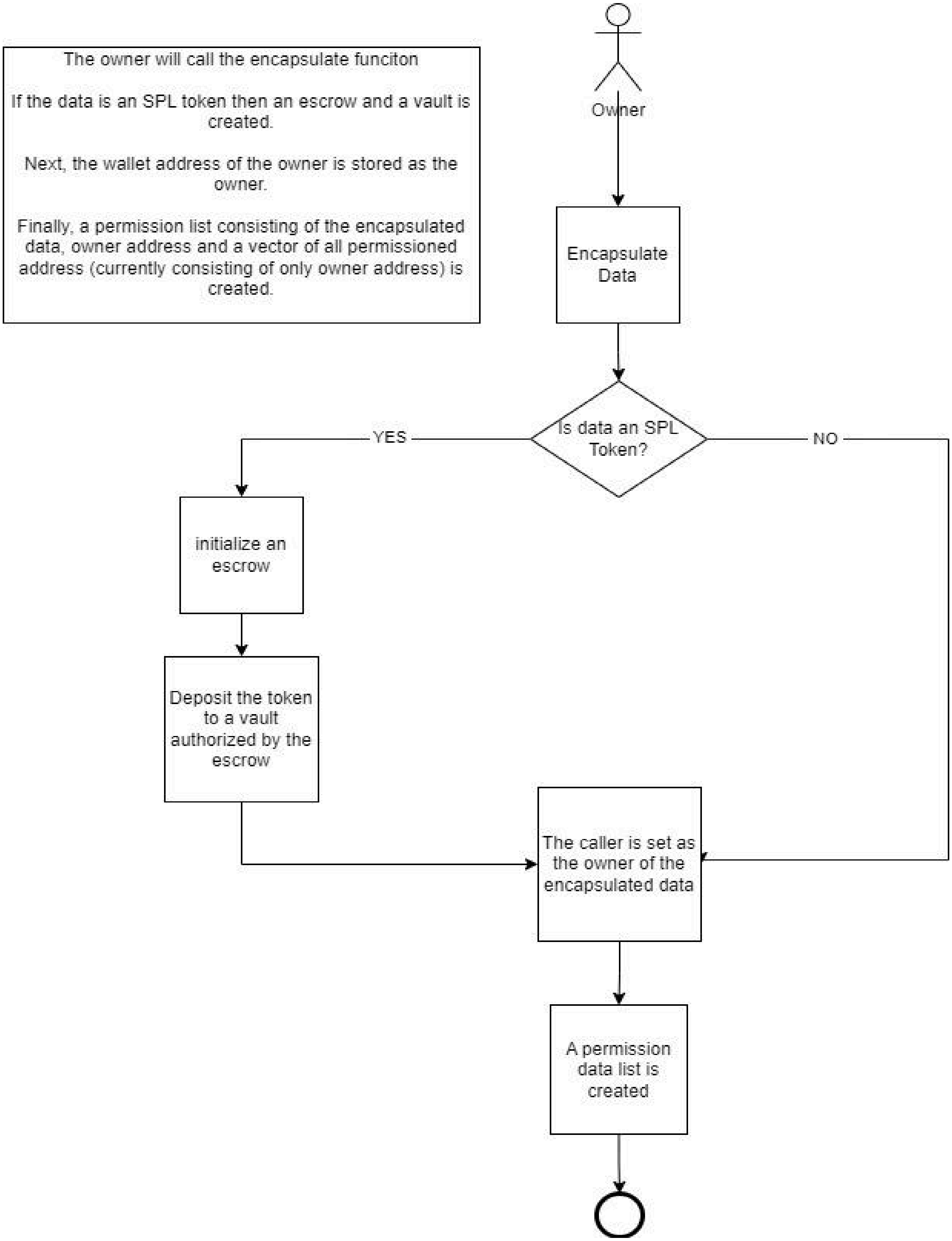
- 9 - accept ownership:
  - The delegated owner will call the function.

- 10 - update owner:
  - The platform will check if the caller is the valid delegated owner and if the delegated start time is reached. If both passes, the platform transfers the ownership.
  - If the input wallet exists then its role is updated and if it does not exist then it is added to the list as the new owner.
  - The role of the original owner is updated to the admin role.

- 11 - edit data:
  - If textual data, the owner or the admin includes the new data as argument.
  - If token, the owner or the admin can add more token or remove token to a different wallet. For this purpose, the caller will call specific functions.

- 12 - transfer if required:
  - If transferring tokens is initiated, the system will transfer the tokens from vault to designated wallet.

Encapsulation



The owner will call the encapsulate funciton

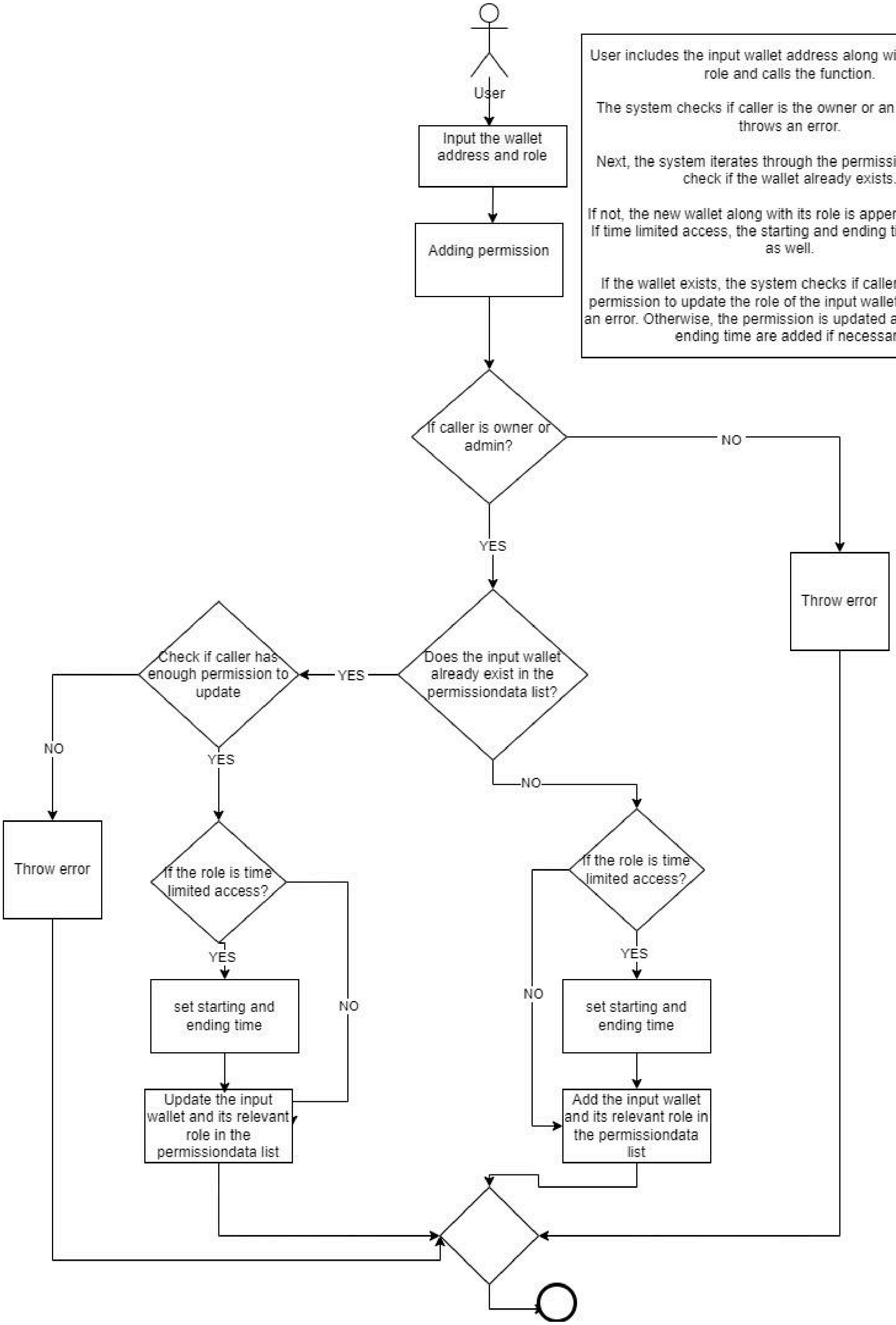
If the data is an SPL token then an escrow and a vault is created.

Next, the wallet address of the owner is stored as the owner.

Finally, a permission list consisting of the encapsulated data, owner address and a vector of all permissioned address (currently consisting of only owner address) is created.

Owner
Encapsulate Data
Is data an SPL Token?
YES
NO
initialize an escrow
Deposit the token to a vault authorized by the escrow
The caller is set as the owner of the encapsulated data
A permission data list is created

Add Permissions


User
Input the wallet address and role
Adding permission
If caller is owner or admin?
NO
Throw error
YES
Does the input wallet already exist in the permissiondata list?
YES
Check if caller has enough permission to update
NO
Throw error
YES
If the role is time limited access?
YES
set starting and ending time
NO
Update the input wallet and its relevant role in the permissiondata list
NO
If the role is time limited access?
NO
YES
set starting and ending time
Add the input wallet and its relevant role in the permissiondata list

User includes the input wallet address along with the relevant role and calls the function.

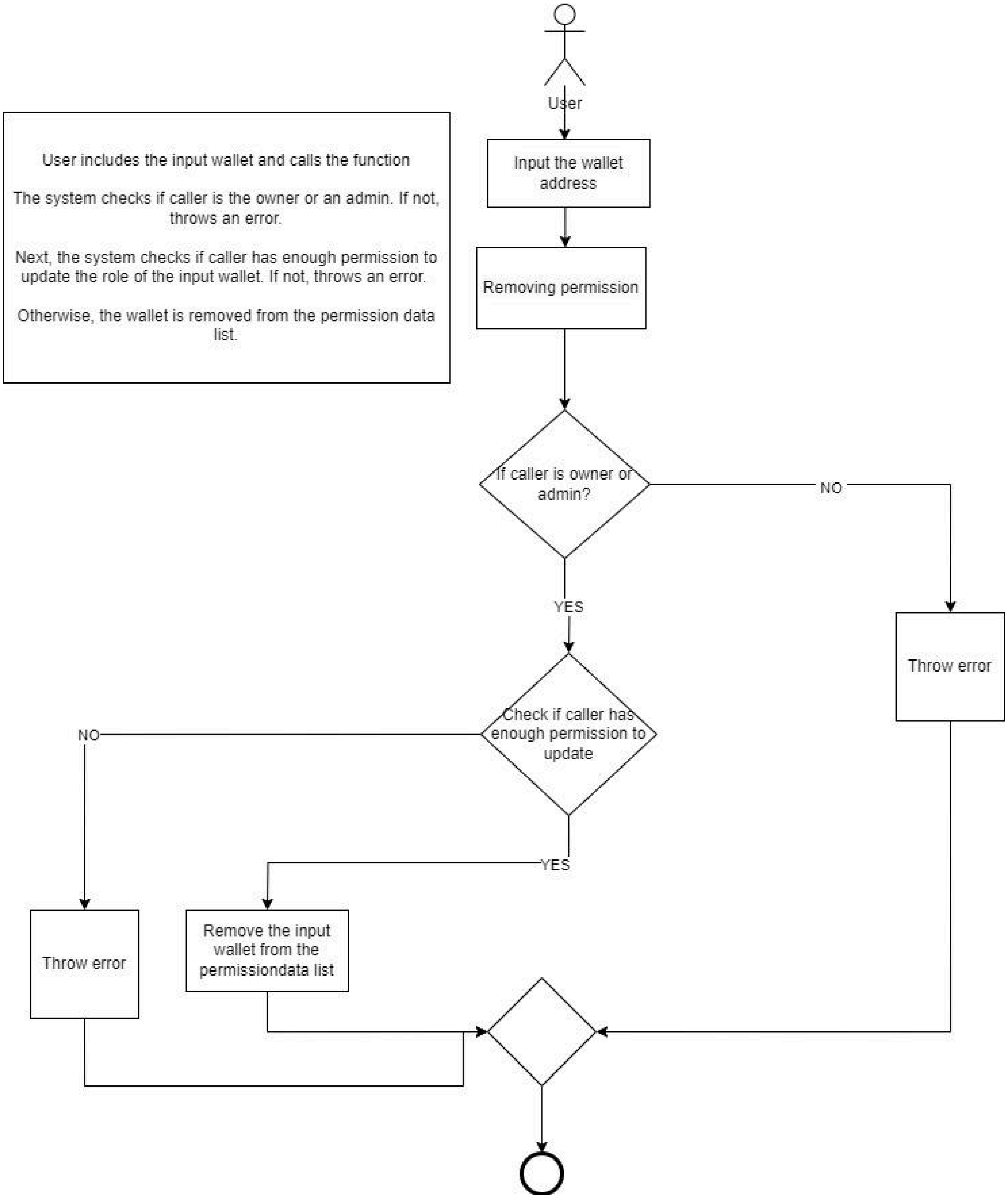
The system checks if caller is the owner or an admin. If not, throws an error.

Next, the system iterates through the permission data list to check if the wallet already exists.

If not, the new wallet along with its role is appended to the list. If time limited access, the starting and ending time are added as well.

If the wallet exists, the system checks if caller has enough permission to update the role of the input wallet. If not, throws an error. Otherwise, the permission is updated and starting and ending time are added if necessary

Remove Permissions



User includes the input wallet and calls the function

The system checks if caller is the owner or an admin. If not, throws an error.

Next, the system checks if caller has enough permission to update the role of the input wallet. If not, throws an error.

Otherwise, the wallet is removed from the permission data list.

User
Input the wallet address
Removing permission
If caller is owner or admin?
NO
Throw error
YES
Check if caller has enough permission to update
NO
YES
Throw error
Remove the input wallet from the permissiondata list

Transfer Ownership

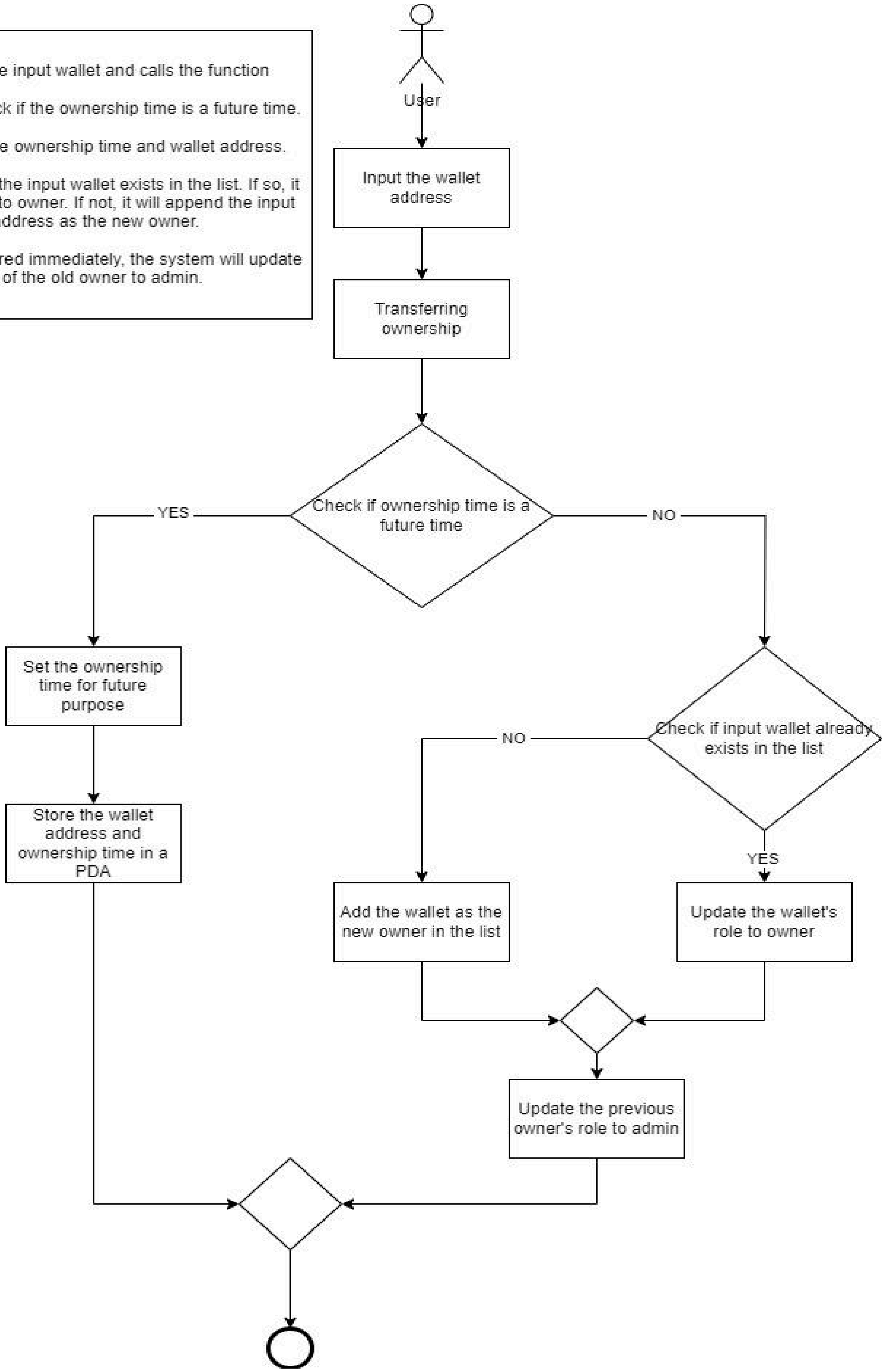
User includes the input wallet and calls the function

The system will check if the ownership time is a future time.

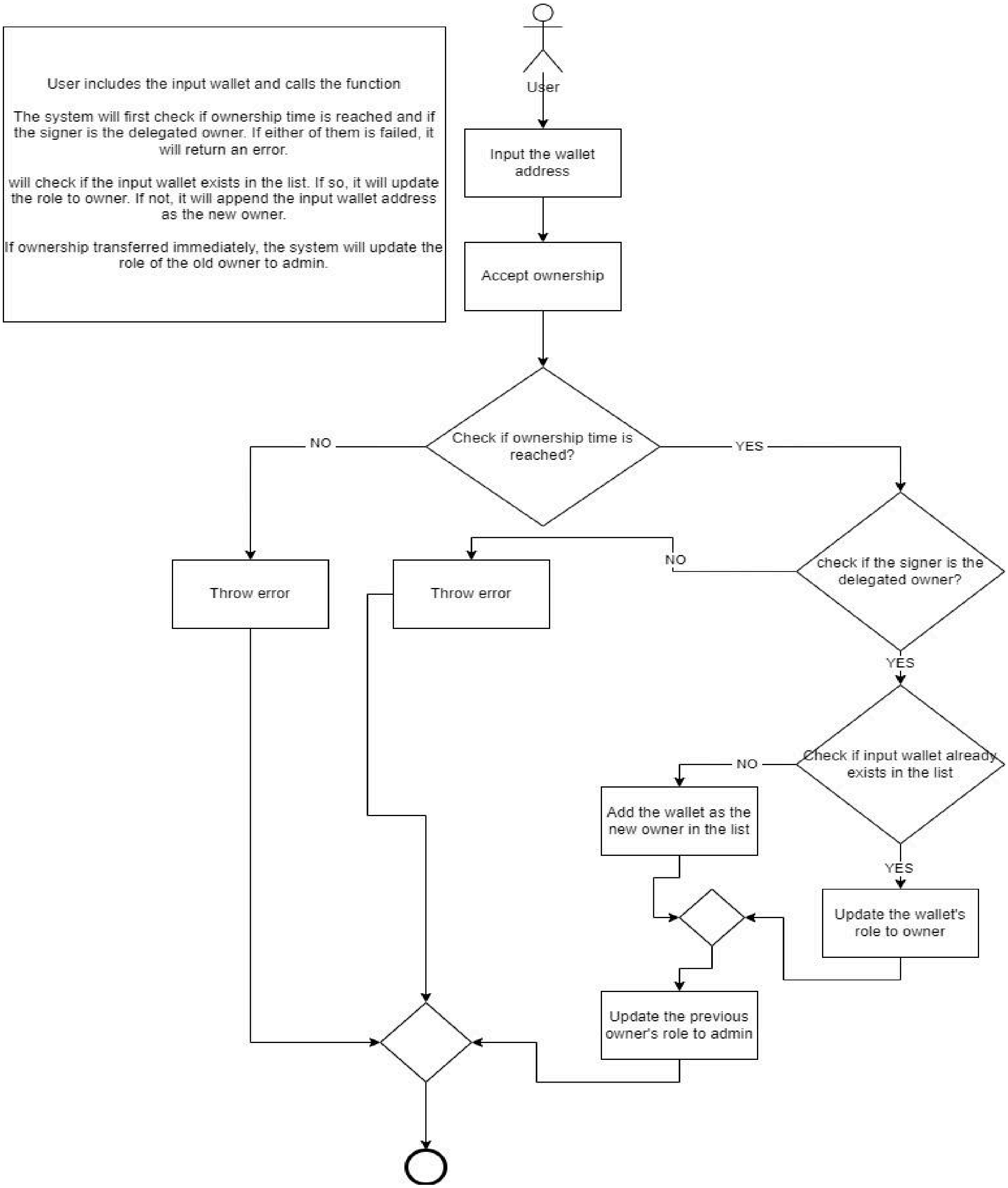
If so, it will store the ownership time and wallet address.

If not, it will check if the input wallet exists in the list. If so, it will update the role to owner. If not, it will append the input wallet address as the new owner.

If ownership transferred immediately, the system will update the role of the old owner to admin.


User
Input the wallet address
Transferring ownership
Check if ownership time is a future time
YES
NO
Set the ownership time for future purpose
Store the wallet address and ownership time in a PDA
Check if input wallet already exists in the list
NO
YES
Add the wallet as the new owner in the list
Update the wallet's role to owner
Update the previous owner's role to admin

Accept Ownership



User includes the input wallet and calls the function

The system will first check if ownership time is reached and if the signer is the delegated owner. If either of them is failed, it will return an error.

will check if the input wallet exists in the list. If so, it will update the role to owner. If not, it will append the input wallet address as the new owner.

If ownership transferred immediately, the system will update the role of the old owner to admin.

User
Input the wallet address
Accept ownership
Check if ownership time is reached?
NO
YES
Throw error
Throw error
NO
check if the signer is the delegated owner?
YES
Check if input wallet already exists in the list
NO
Add the wallet as the new owner in the list
YES
Update the wallet's role to owner
Update the previous owner's role to admin

Edit Data

User starts editing data

Checks if user has access (if time limited, check if current time within the start and end time). If no, throw error.

Check if token.

If not, update the data.

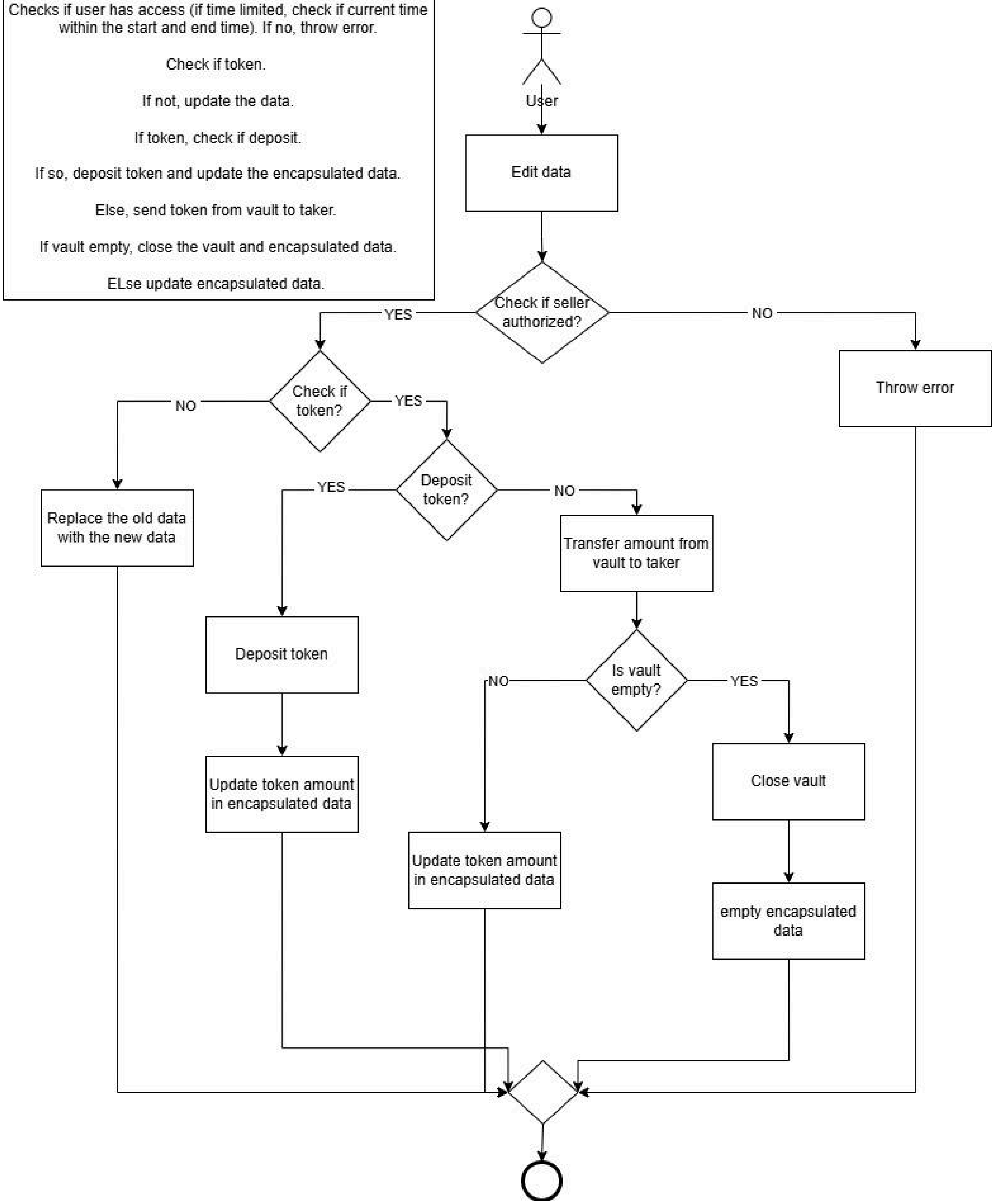
If token, check if deposit.

If so, deposit token and update the encapsulated data.

Else, send token from vault to taker.

If vault empty, close the vault and encapsulated data.

ELse update encapsulated data.


User
Edit data
Check if seller authorized?
YES
NO
Throw error
Check if token?
NO
YES
Replace the old data with the new data
Deposit token?
YES
NO
Deposit token
Transfer amount from vault to taker
Is vault empty?
NO
YES
Update token amount in encapsulated data
Close vault
Update token amount in encapsulated data
empty encapsulated data

End to End Overview

1 - Owner will encapsulate data.

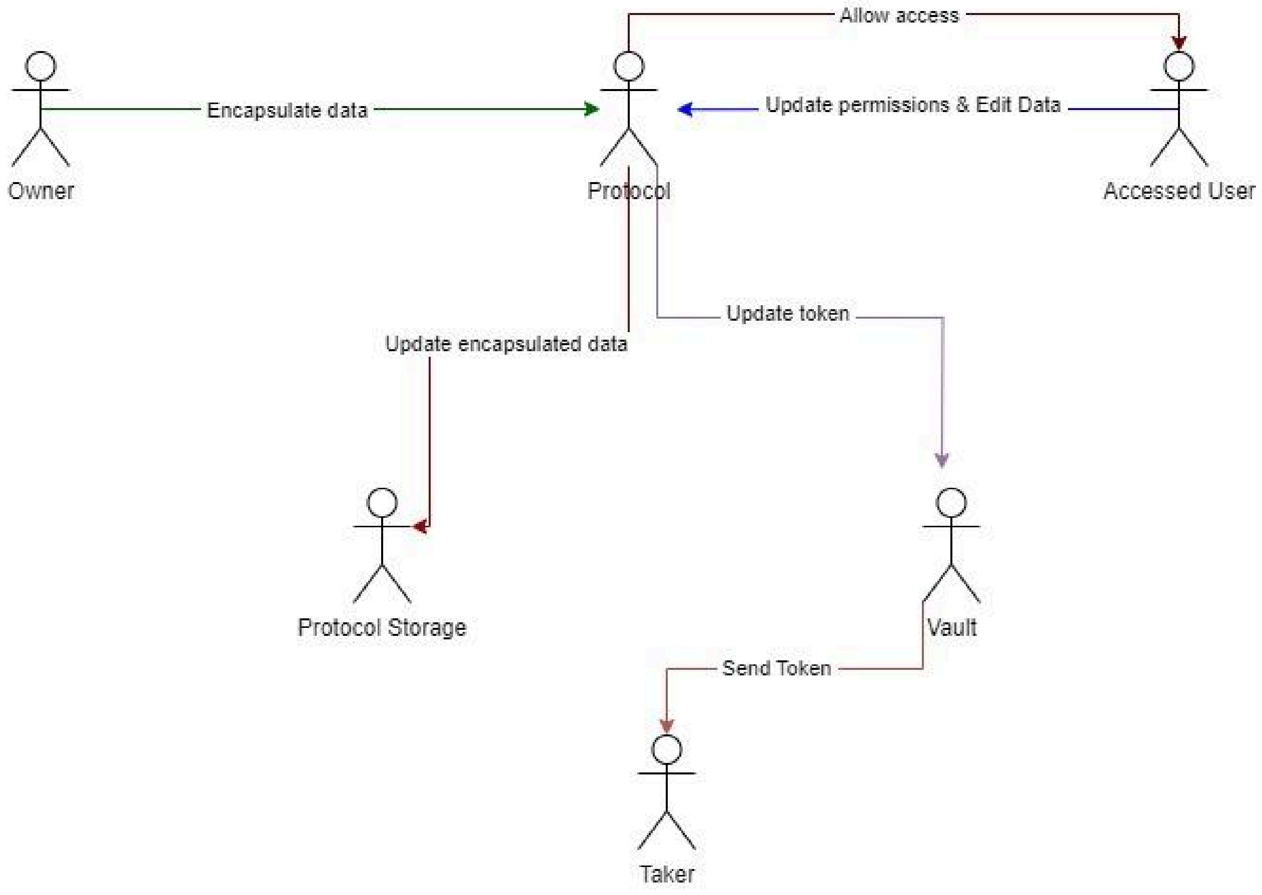
2 - Data is encapsulated and stored in storage by the protocol.

3 - Protocol will allow access to designated users.

4 - Designated users can update permissions and edit data.

5 - Token data will be updated in a vault.

6 - Vault can send data to taker.


Allow access
Owner
Encapsulate data
Protocol
Update permissions & Edit Data
Accessed User
Update token
Update encapsulated data
Protocol Storage
Vault
Send Token
Taker