I. Introduction

EtherID is a decentralized application (dApp) that provides DNS-like services for Ethereum wallets. It allows users to register human-readable domain names that map to Ethereum addresses, making it easier to send and receive cryptocurrency without dealing with long, complex addresses.

II. Related Works

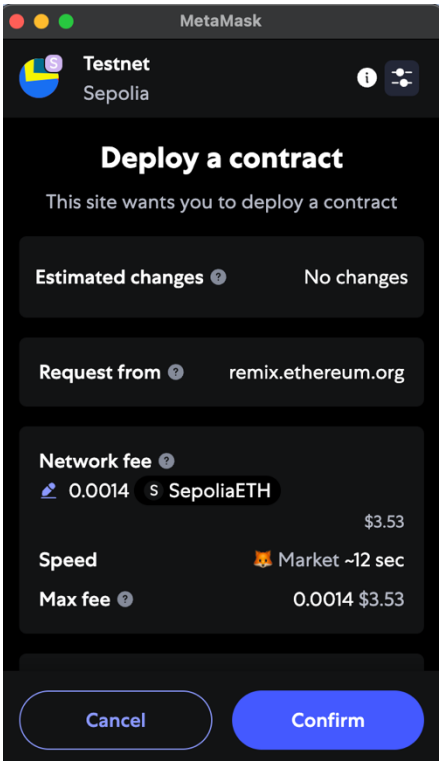
EtherID is similar to Ethereum Name Service (ENS), providing readable domain names for Ethereum addresses. Both systems map complex blockchain addresses to memorable names, but EtherID offers a simpler, more focused implementation with direct payment capabilities using web3.0 approach.

III. Core Features

1. Domain Registration & Management System
    - By borrowing the concept of Domain Name System into blockchain applications, users can register domain names as they desire.
    - Once the domain is mapped to a wallet address, the domain will be locked to the corresponding wallet address until the domain is available again.
    - Easy administration can be done on the Admin Panel.
    - The register domain can be used to provide a shortcut for payment transfer.
2. Registration Fee Collection
    - As each domain require certain amount of registration fees to register, all accumulated registration fees can be collected by the contract owner.
3. Intuitive User Interface
    - Simple and introduction UI leading user to use the dApp.
    - Users can connect MetaMask wallet to use the dApp.
    - The dApp is dynamically updating, i.e. real time registration fee, sensitive to wallet changing and contract owner only Admin Panel.
4. Level of Protection
    - The functions defined in the smart contract is prohibited to different level of permissions to increase security level.
    - Functions will be only called by the contract owner (a.k.a admin)
    - Protection in terms of accessibility is also introduced, before entering functions body, several check will be performed to avoid redundant contract calling.

IV. Testing

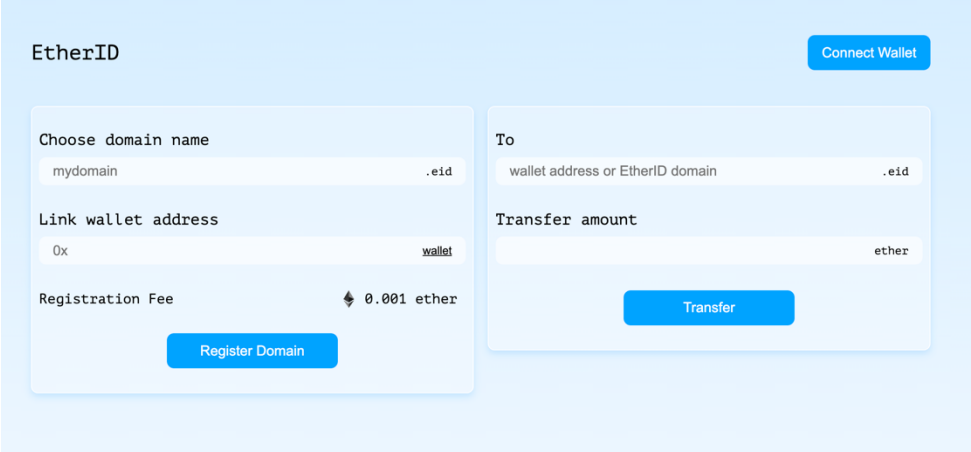
Test #1 – Deploy contract on Sepolia testnet with Remix.



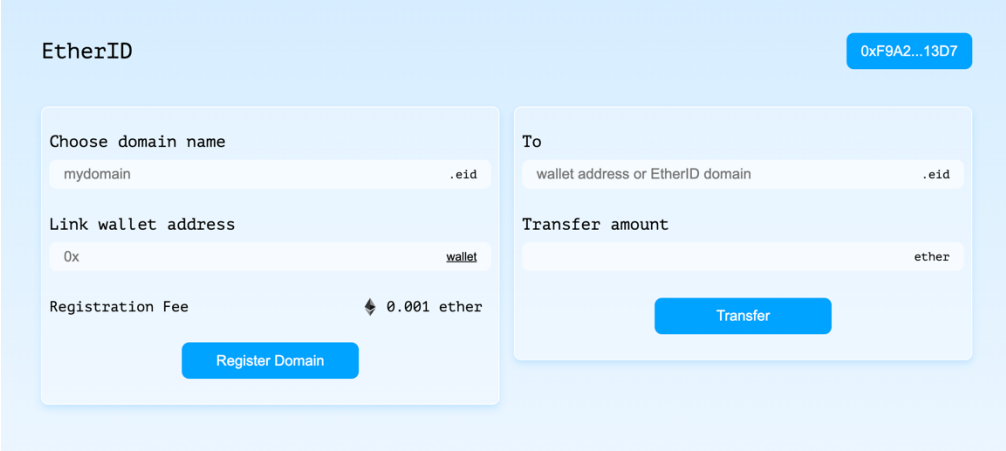
1. Contract Owner: 0xb56d84038e4ad05081a08874c950599d51e553f2
2. Contract Address: 0xB1bCadA08aD3C30de7c89201Ac82fA786175d265
https://sepolia.etherscan.io/address/0xB1bCadA08aD3C30de7c89201Ac82fA786175d265
3. The contract owner will be denoted by **OWNER** for later.

Test #2 – Before and after wallet connection.

(before)



(after)

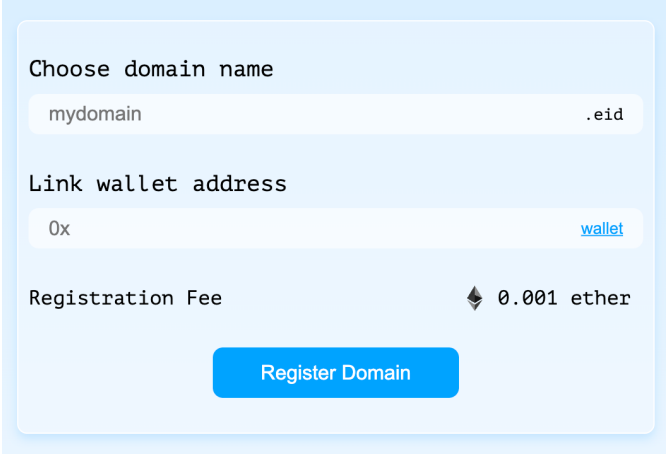


1. Using another account (**TEST**) for testing purpose.
2. **TEST** Address : 0xF9A2b75bA09FdD8d39F2f1f392a96297588013D7
3. **TEST** implies normal users as they are not the contract owner.
4. Note that the “Connect Wallet” button already represented by the user wallet address.
5. Note that the registration fee is dynamically get from the smart contract, i.e. 0.001 ether registration fee.

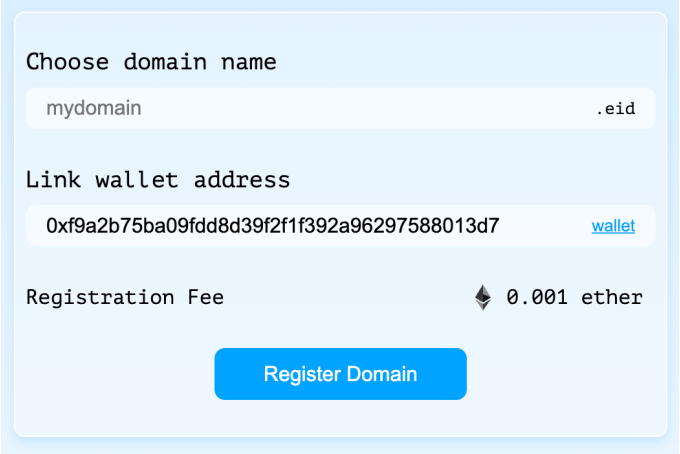
Test #3 – Register a domain for TEST.

A. The copy wallet button helps users to load their connected wallet address.

(before)

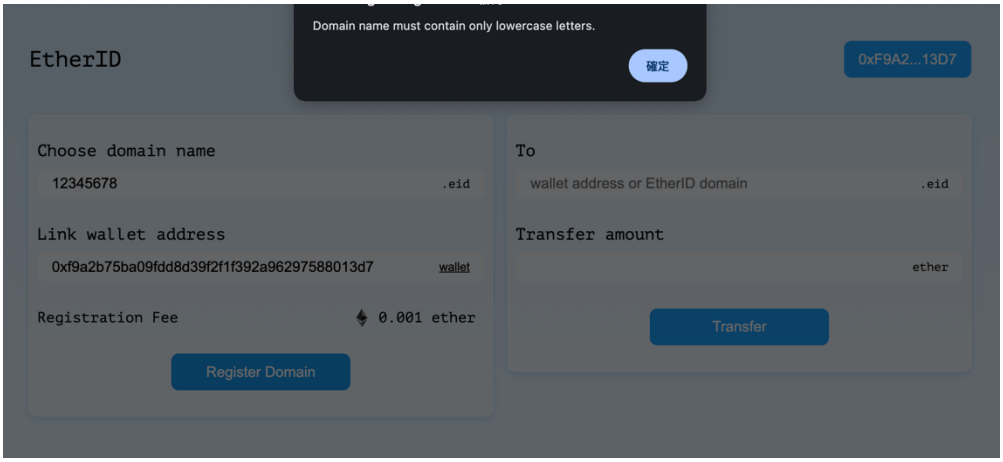


(after)

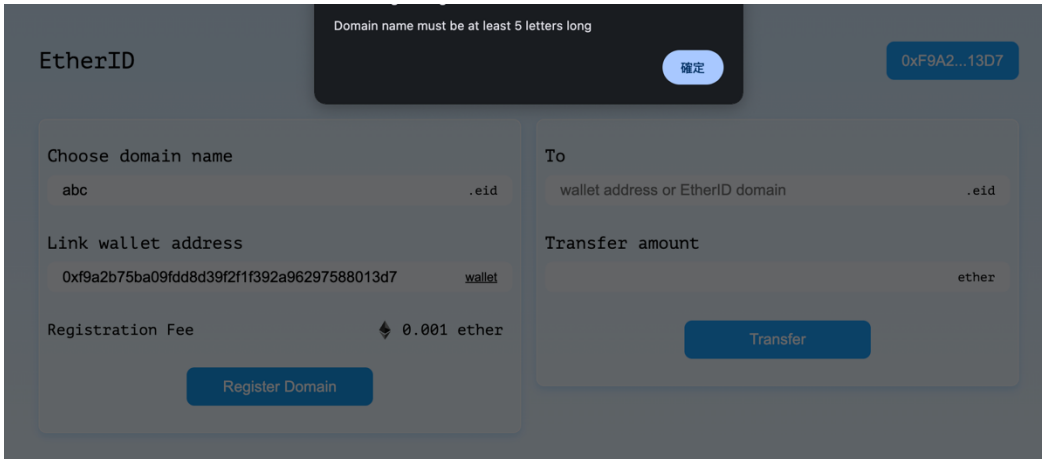


B. Domain name will be constrained to lowercase letters only.

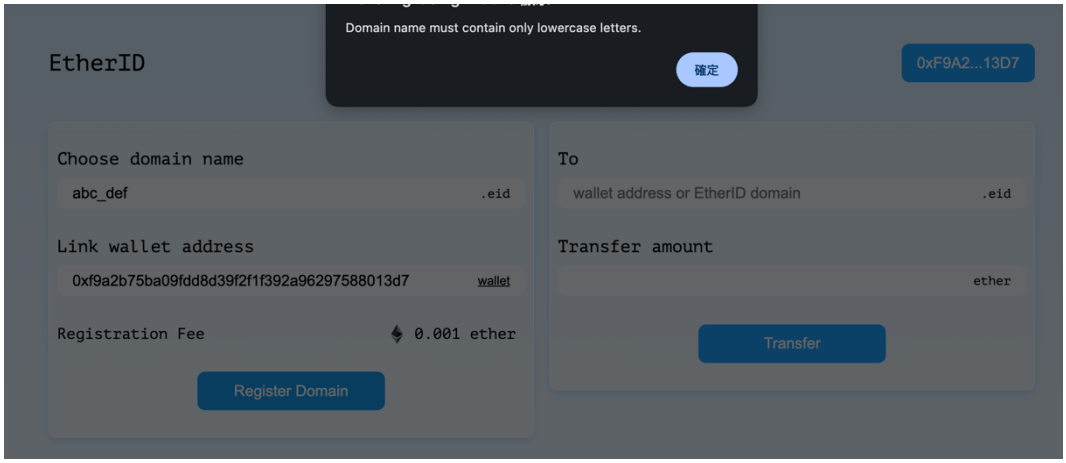
(included numbers)



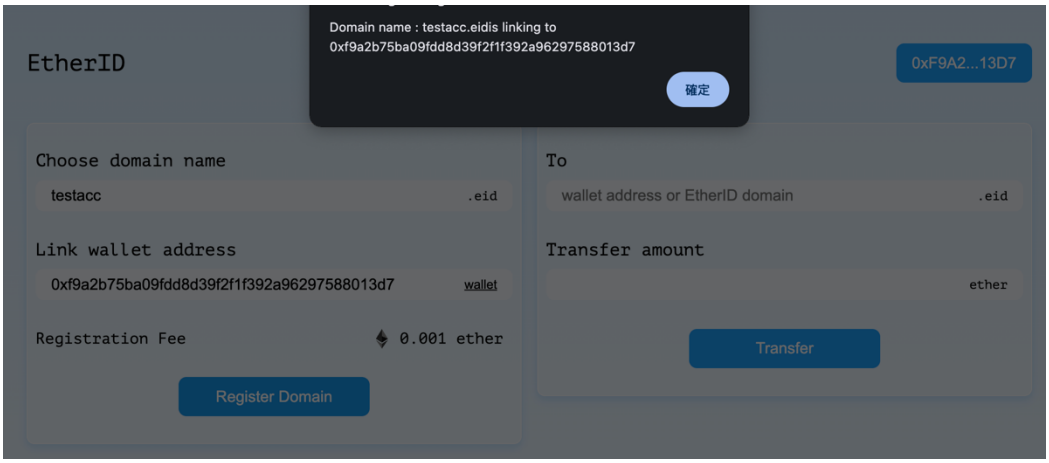
(less than 5 letters)

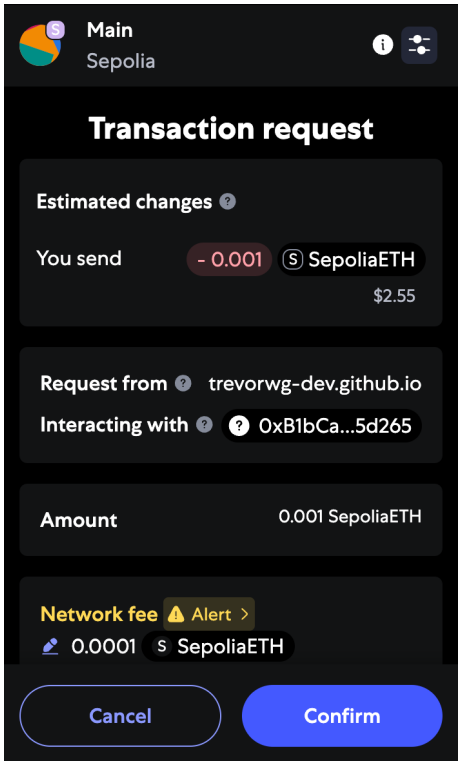


(included invalid characters)



(acceptable domain name)



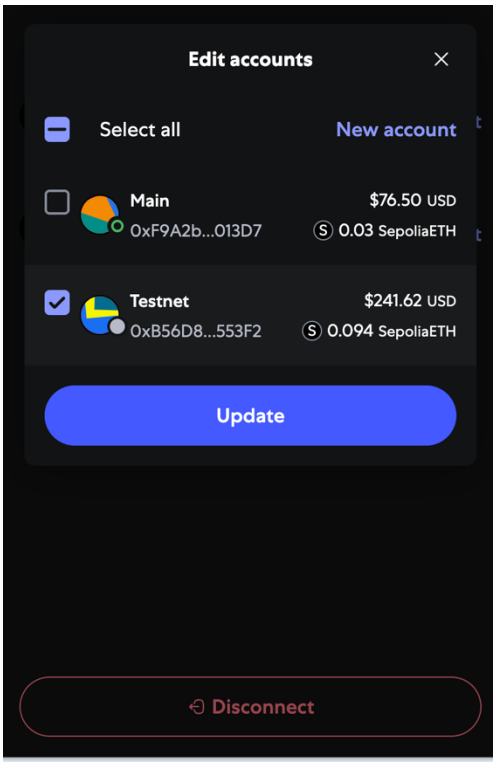


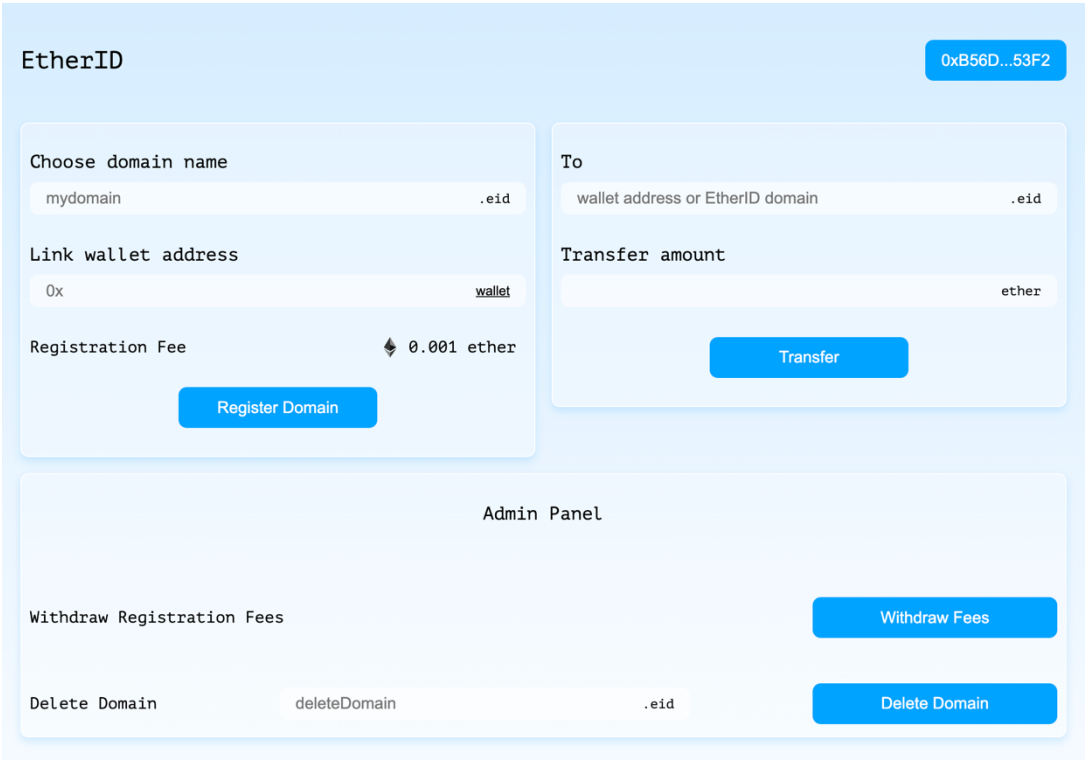
1. The registration transaction: 0x782b00b8fbb16bcaa3e8076afb423a2a10b5808ef0972dc7a98937f1ac38d47c

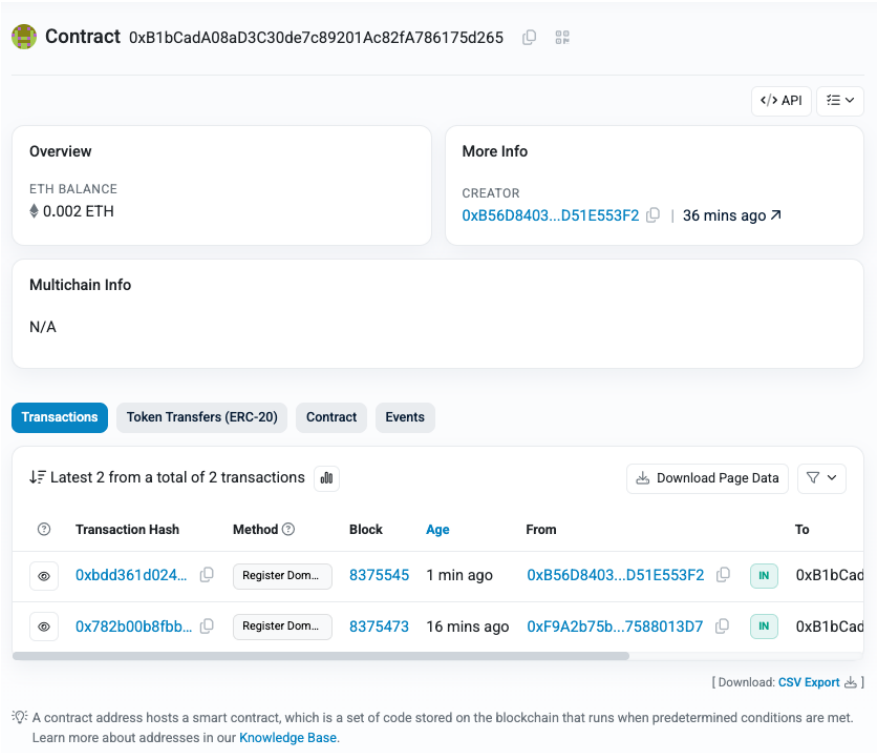
https://sepolia.etherscan.io/tx/0x782b00b8fbb16bcaa3e8076afb423a2a10b5808ef0972dc7a98937f1ac38d47c

2. Exactly the required registration fee is paid to the smart contract.

Test #4 – Update the front-end when wallet address is changed.





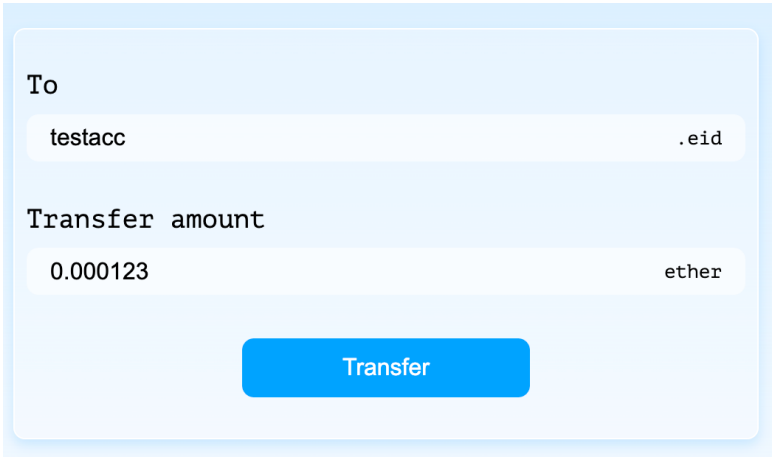


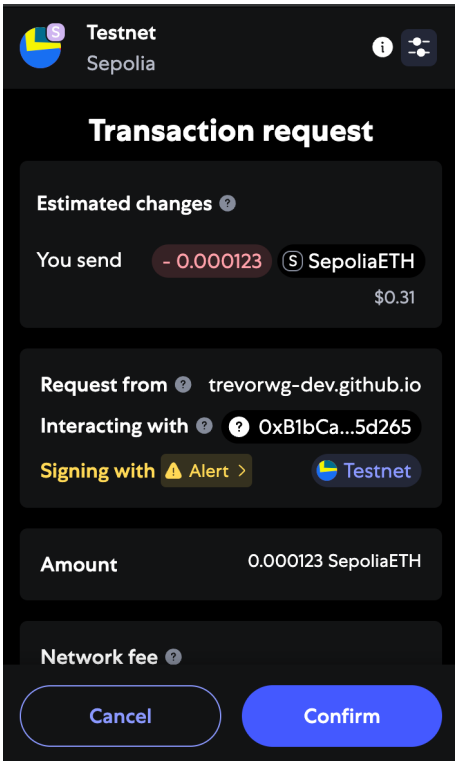
1. When the wallet changed (from **TEST** to **OWNER**) both the right-corner button changed to represent the **OWNER** address.
2. Since this address is the contract owner, the admin panel is rendered.
3. Register domain for **OWNER** for later testing, such that **TEST** is using "testacc.eid" as domain name, while **OWNER** is using "admin.eid".
4. The OWNER domain name is register to the contract: 0xbdd361d0245a3115305b46901b5d927b8c9b24586d6a1f765f5afdda6f6580c4

https://sepolia.etherscan.io/tx/0xbdd361d0245a3115305b46901b5d927b8c9b24586d6a1f765f5afdda6f6580c4

5. The contract balance is 0.002 ETH as expected.

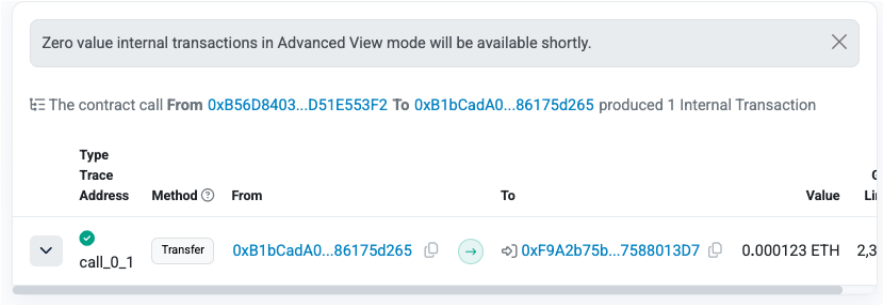
Test #5 – Transfer 0.000123 ETH from **OWNER** to **TEST** though domain



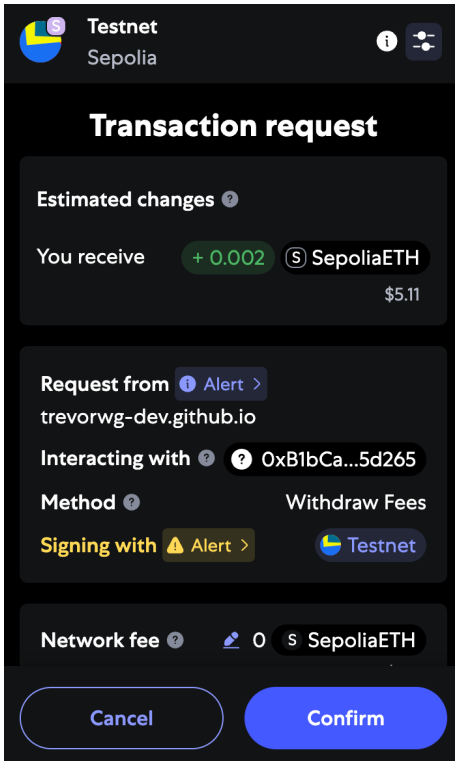


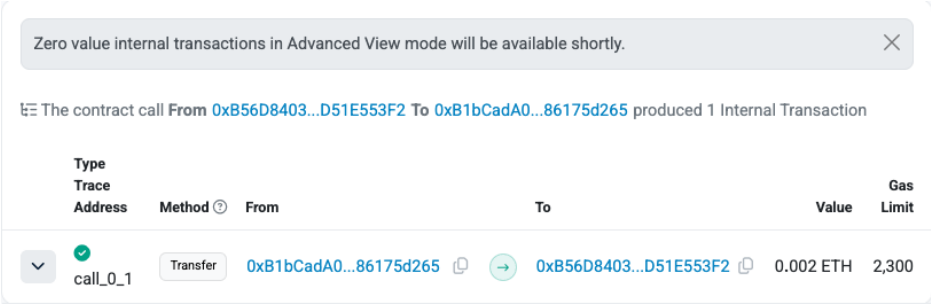
1. The **OWNER** to contract transaction: https://sepolia.etherscan.io/tx/0x6cbacfdd3ffb7ab81ce8a06f8037e58f77dea562979c8a42763e4ea24015c959
2. The exact value of 0.000123 ETH will be sent to the contract, then the contract will further transfer to target user.

3. The contract to **TEST** *internal* transaction:



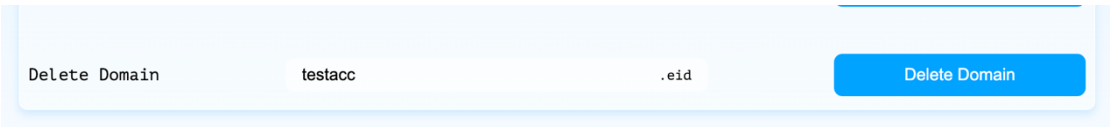
Test #6 – **OWNER** Withdraw registration fees

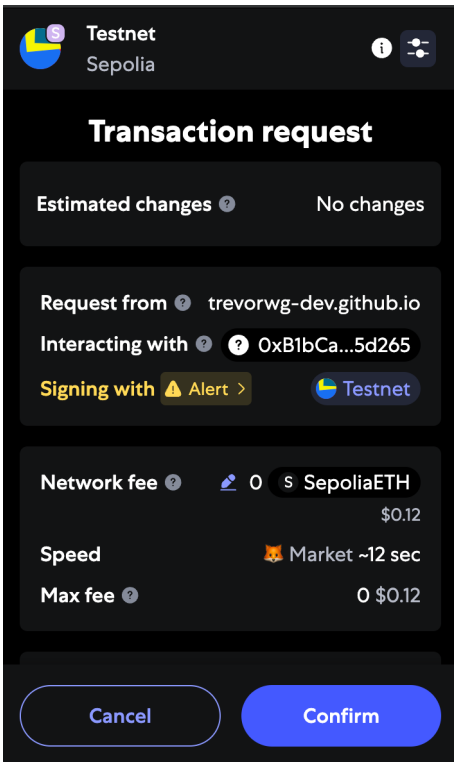




1. As expected, we register 2 domain names, so the contract will have 0.002 ETH, and now we are collecting back using the UI in admin panel.
2. The transaction address: https://sepolia.etherscan.io/tx/0xd9d9bc40cc8ed87ed3b0beae7ce204eb9a9176e4c661f40d099e36673d52ccb9

Test #7 – **OWNER** delete existing domain name





1. The mapping for “testacc.eid” to the **TEST** address will be replaced by 0s, thus the mapping is deleted.
2. The transaction address: https://sepolia.etherscan.io/tx/0x60f60aeddc05120bcc4d8294a3d0acf6e42bc8964d4b3ddcb1edfb7ae15c6aed

V. Beyond Development

1. Better UI / Notification system to notice user to wait for next action.
2. Introduce user-side domain management system, i.e. they can manage how long they want to “rent” the domain name and perhaps can renew or drop the domain name when they desire.
3. Although, this is a small project using dApp and blockchain concept, backend server to handle user responsive will be better, rather than let the smart contract or front-end logic handling.