Testing the Fuego DApp using Truffle

Overview

The **Fuego DApp** allows users to mint **2000 Fuego tokens (FGO) once** and transfer tokens to other users. The contract ensures that each user can only mint once, and only the owner can manually mint additional tokens.

This document provides instructions on testing the DApp using **Truffle**, ensuring all functionalities work as expected.

Read the full documentation here:
https://github.com/dkokonkwo/assignment_3_DApp/blob/main/README.md

Prerequisites

Ensure the following are installed before proceeding:

- **Node.js** (>=16.x) → [Download Here]()
- **Truffle** → `npm install -g truffle`
- **Ganache** (for local blockchain testing) → [Download Here]()
- **MetaMask** (browser extension) → [Download Here]()
- **Ethereum Wallet with Testnet Funds**

1. Setting Up the Test Environment

Clone the repository and install dependencies:

```
git clone https://github.com/dkokonkwo/assignment_3_DApp.git
cd fuegoWallet
npm install
```

Start **Ganache** and configure **Truffle** to use it.

Edit `truffle-config.js` to include the following:

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*",
    },
  },
  compilers: {
    solc: {
      version: "0.8.22",
    },
  },
};
```

2. Running Tests with Truffle

Use **Truffle's built-in testing framework** to ensure the contract behaves as expected.

Compile the Smart Contract

```
truffle compile
```

Deploy to Local Blockchain

```
truffle migrate --network development
```

Run Test Cases

```
truffle test
```

3. Test Cases Explained

Test 1: Contract Deployment

- Ensures the contract deploys successfully.
- Verifies that the deployed contract instance is not `undefined`.

Test 2: Minting Tokens (Positive Case)

- Calls **`mintOnce()`** from an account.
- Checks if the account balance updates correctly (should be **2000 FGO**).

Test 3: Prevent Multiple Mints (Negative Case)

- Calls **mintOnce()** again from the same account.
- Should **fail** with **"Token already minted."**

Test 4: Owner Minting Additional Tokens

- The contract owner mints tokens to a different address.
- Checks if the balance updates accordingly.

Test 5: Prevent Unauthorized Minting

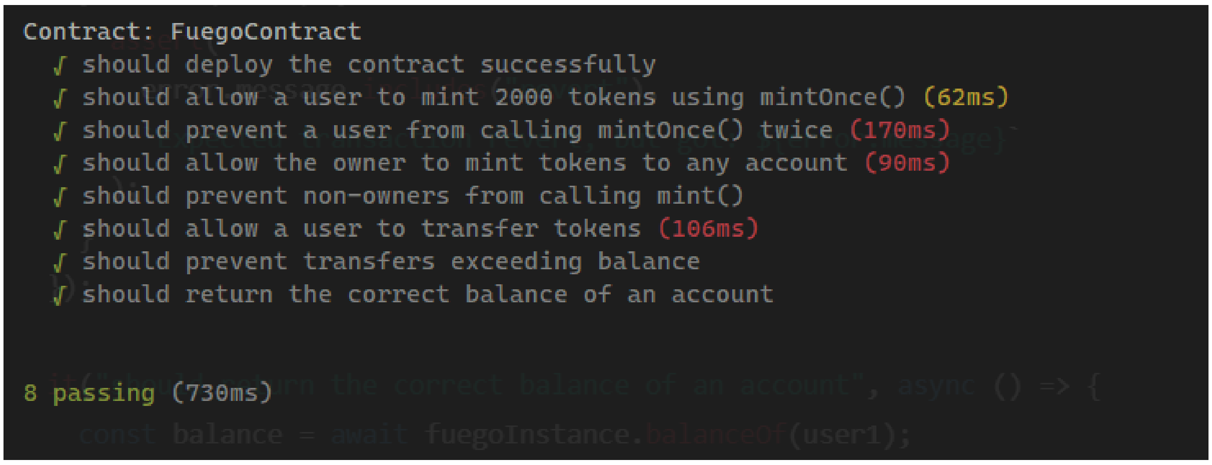
- A non-owner account attempts to call **mint()**.
- Should **fail** with **"Ownable: caller is not the owner"**.

Test 6: Token Transfer

- User transfers **500 FGO** to another account.
- Checks if balances update correctly.

Test 7: Prevent Transfers Exceeding Balance

- A user attempts to transfer more tokens than they own.
- Should **fail** with **"ERC20: transfer amount exceeds balance"**.



```
Contract: FuegoContract
  √ should deploy the contract successfully
  √ should allow a user to mint 2000 tokens using mintOnce() (62ms)
  √ should prevent a user from calling mintOnce() twice (170ms)
  √ should allow the owner to mint tokens to any account (90ms)
  √ should prevent non-owners from calling mint()
  √ should allow a user to transfer tokens (106ms)
  √ should prevent transfers exceeding balance
  √ should return the correct balance of an account


8 passing (730ms)
```

Testing ensures that the Fuego DApp is secure and functions correctly.
If all tests pass, the DApp is ready for deployment on a public blockchain!

How the Fuego DApp Works

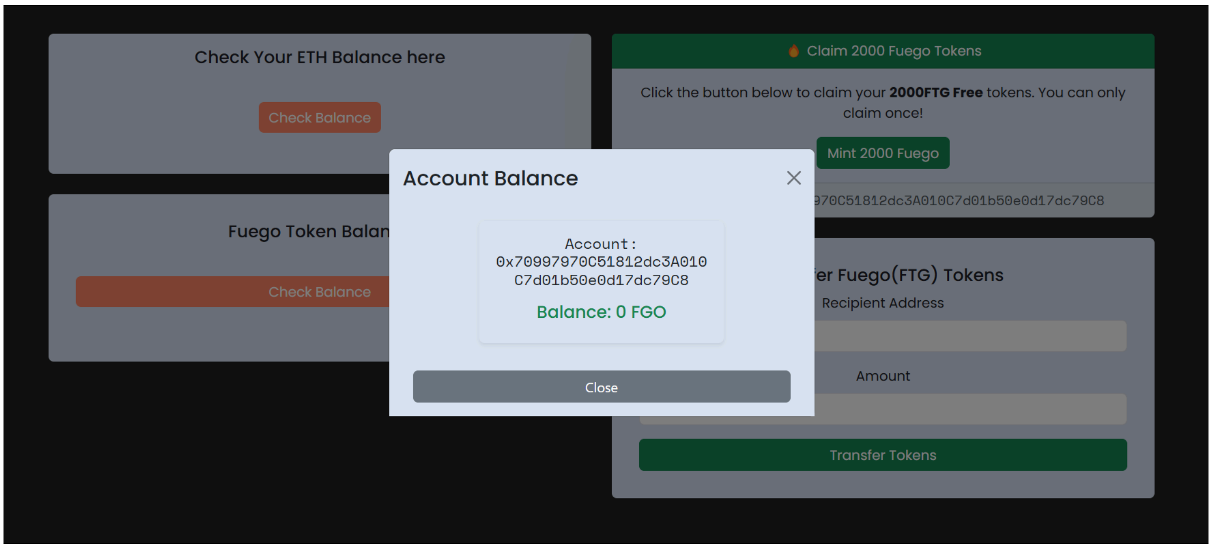
The **Fuego DApp** is a decentralized application that allows users to mint, transfer, and check their balance of **Fuego tokens (FGO)** on the Ethereum blockchain. Below is a step-by-step breakdown of how it functions.

Smart Contract Logic

At the core of the application is the **Fuego Smart Contract**, written in Solidity. It handles the following operations:

Minting Tokens

- Each user can call the `mintOnce()` function **only once** to receive **2000 FGO tokens**.


Account Balance

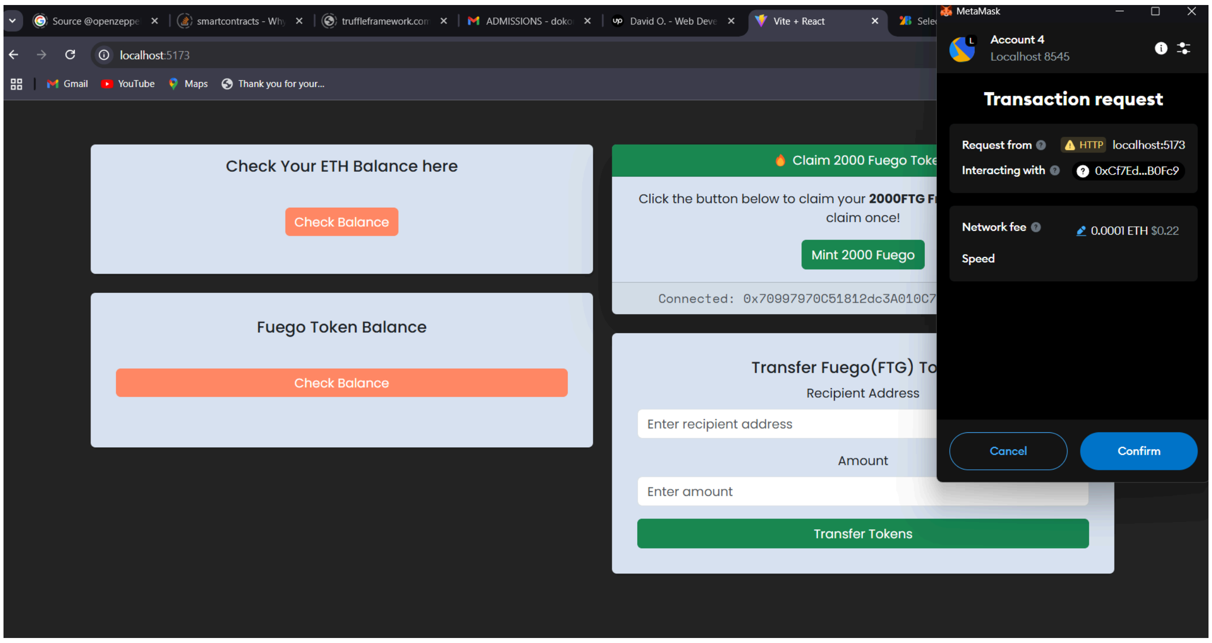
Account:
0x70997970C51812dc3A010C7d01b50e0d17dc79C8

Balance: 0 FGO

Check Your ETH Balance here

Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

- The contract keeps track of which users have already minted using a mapping.


Check Your ETH Balance here

Fuego Token Balance

Click the button below to claim your **2000FTG F** claim once!

Connected: 0x70997970C51812dc3A010C7

Transfer Fuego(FTG) To

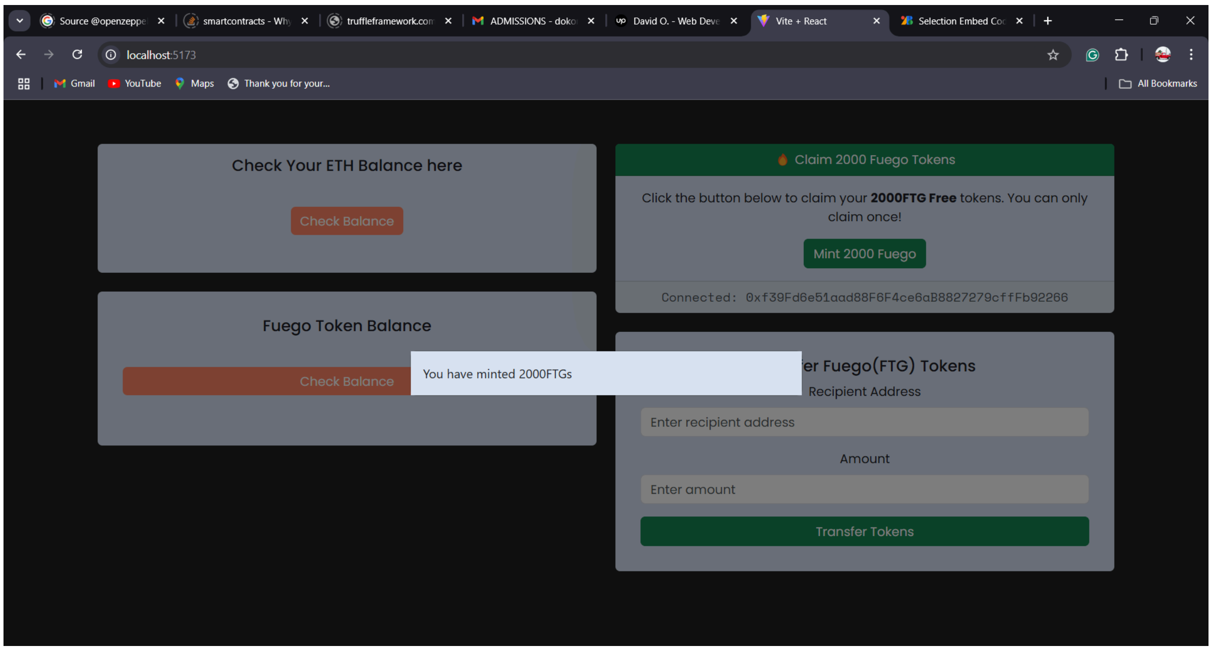
Transaction request

Request from HTTP localhost:5173

Interacting with 0xCf7Ed...B0Fc9

Network fee 0.0001 ETH $0.22

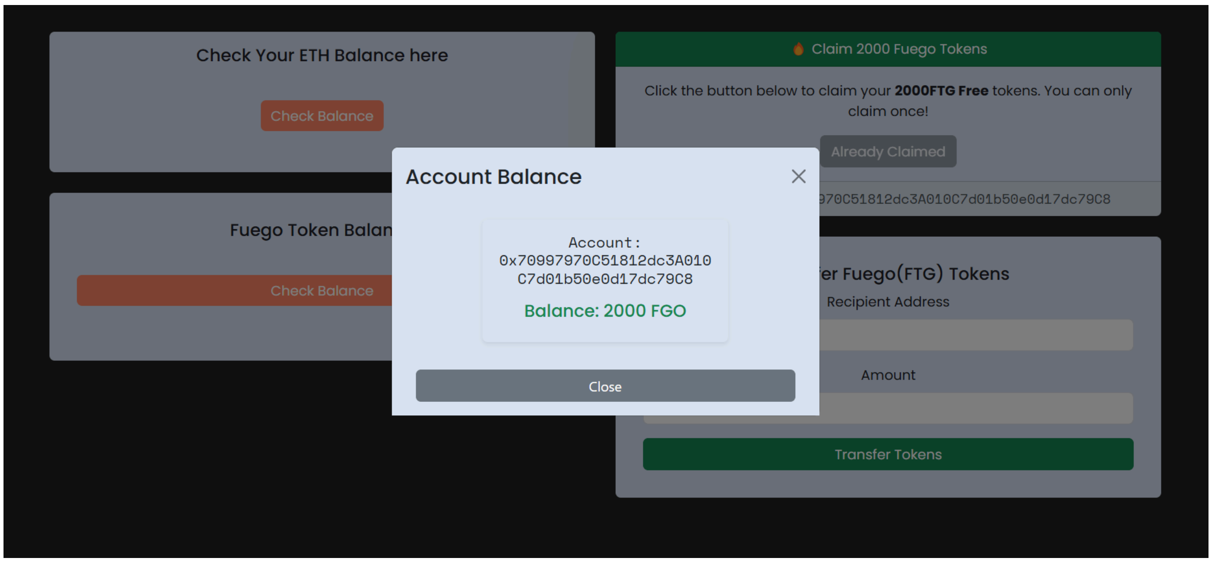
Speed



Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

Connected: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

You have minted 2000FTGs



Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

970C51812dc3A010C7d01b50e0d17dc79C8

Account Balance

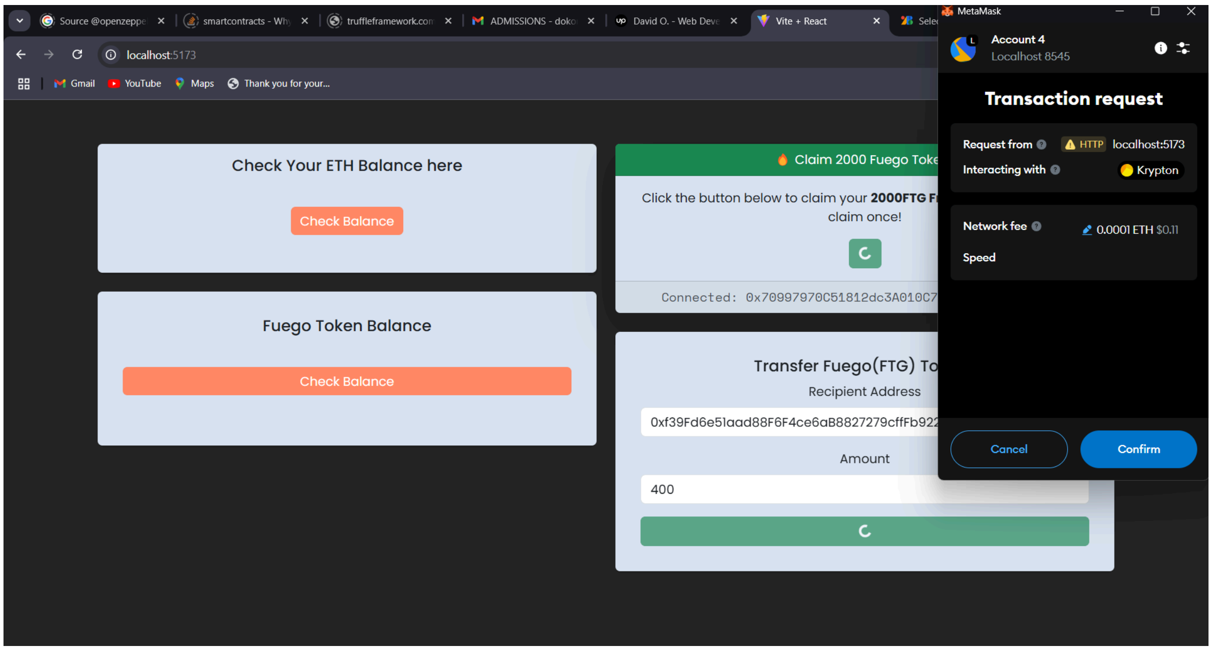
Account:
0x70997970C51812dc3A010C7d01b50e0d17dc79C8

Balance: 2000 FGO

- The owner of the contract can use the `mint(address to, uint256 amount)` function to mint tokens for any user.

Token Transfers

- Users can send **FGO tokens** to others using the `transferTokens(address to, uint256 amount)` function.



Check Your ETH Balance here

Fuego Token Balance

Claim 2000 Fuego Toke

Click the button below to claim your **2000FTG F** claim once!

```
Connected: 0x70997970C51812dc3A010C7
```

Transfer Fuego(FTG) To

Recipient Address

0xf39Fd6e51aad88F6F4ce6aB8827279cffFb922

Amount

400

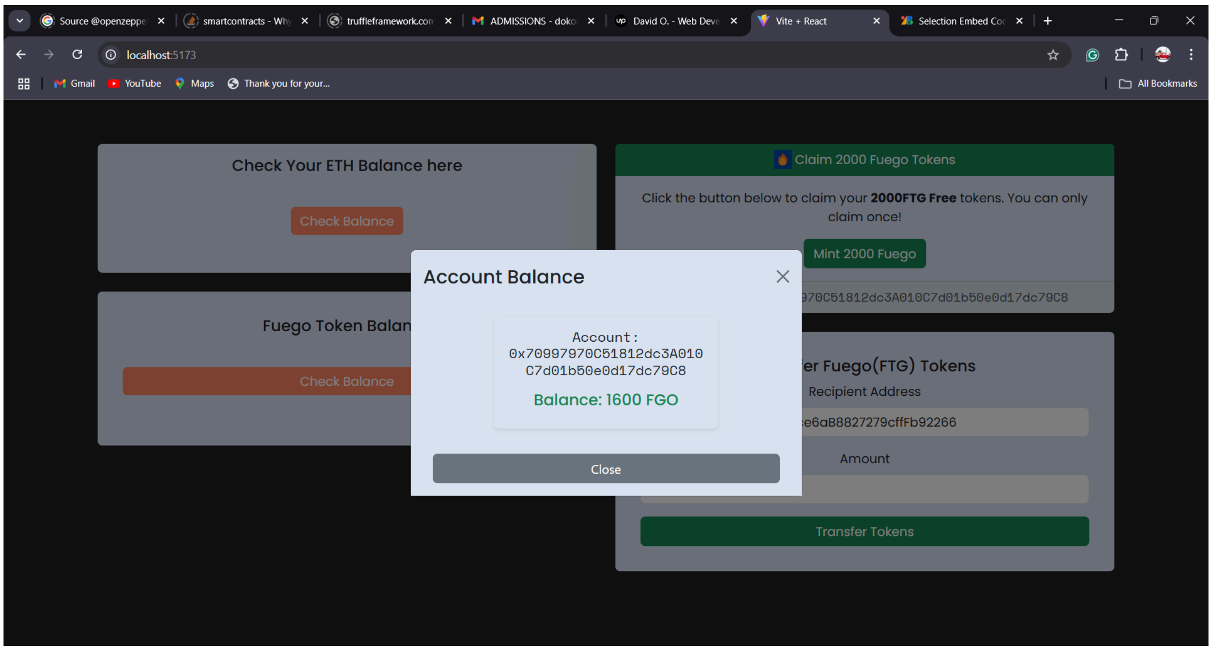
MetaMask

Account 4
Localhost 8545

Transaction request

Request from HTTP localhost:5173
Interacting with Krypton

Network fee 0.0001 ETH $0.11
Speed



Check Your ETH Balance here

Fuego Token Balan

Claim 2000 Fuego Tokens

Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

```
970C51812dc3A010C7d01b50e0d17dc79C8
```

er Fuego(FTG) Tokens

Recipient Address

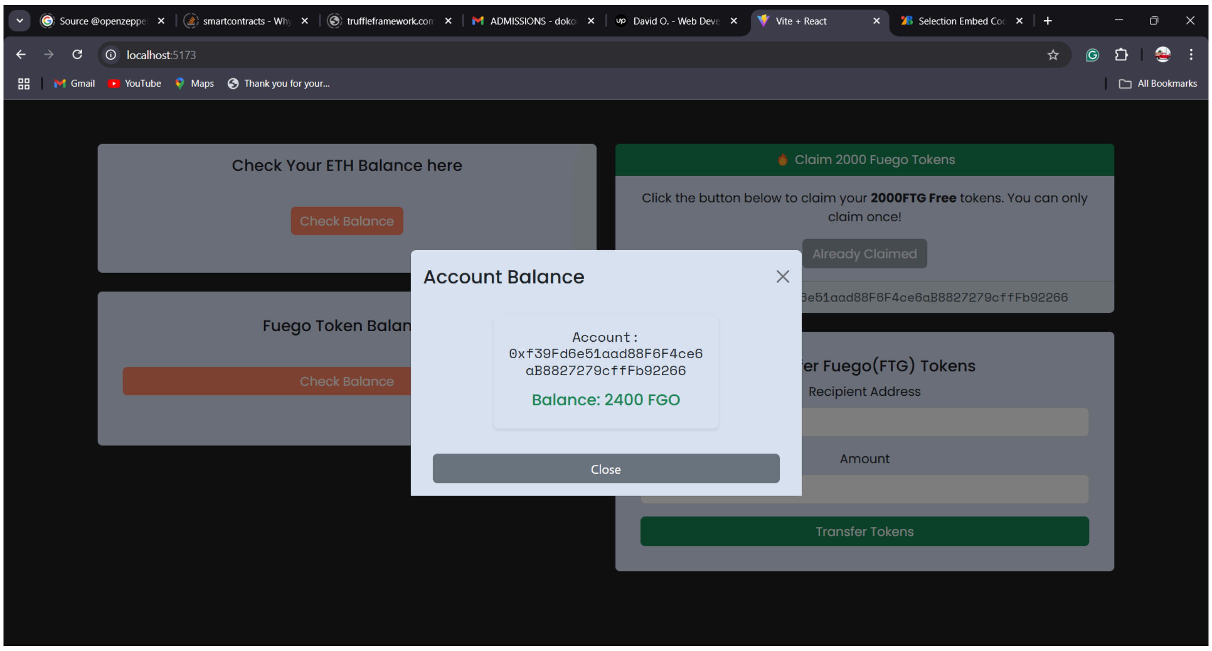
e6aB8827279cffFb92266

Amount

Account Balance

```
Account:
0x70997970C51812dc3A010
C7d01b50e0d17dc79C8
```

Balance: 1600 FGO



Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

Account Balance

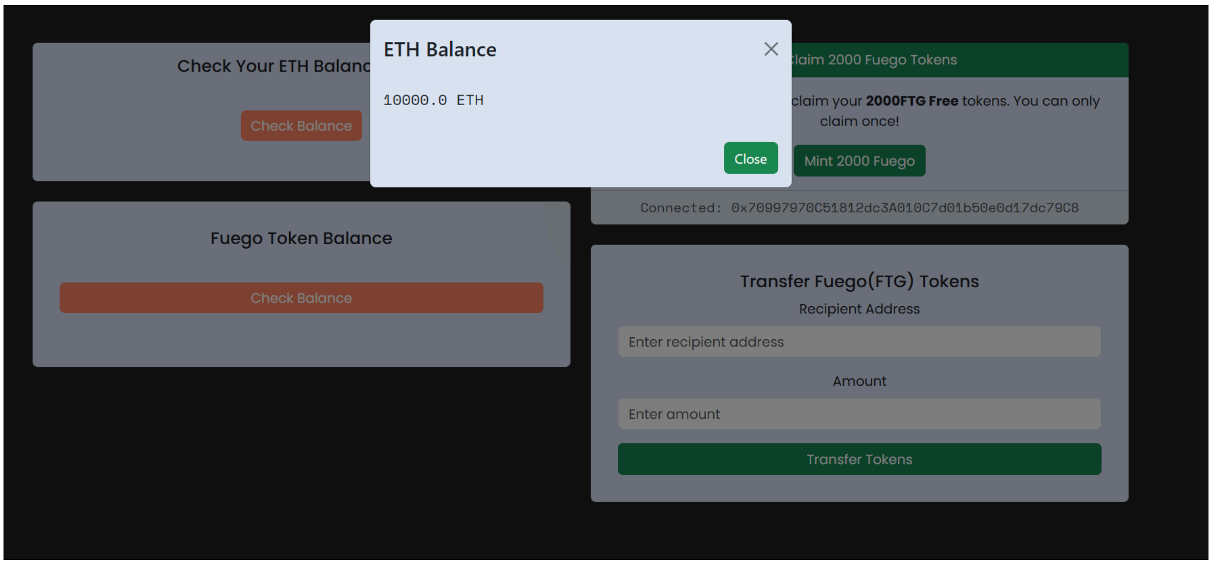
```
Account:
0xf39Fd6e51aad88F6F4ce6
aB8827279cffFb92266
```

Balance: 2400 FGO

- The contract ensures that users **cannot transfer more tokens than they own**.

Checking Balance

- Any user can check their Fuego balance by calling `checkBalance(address account)`.

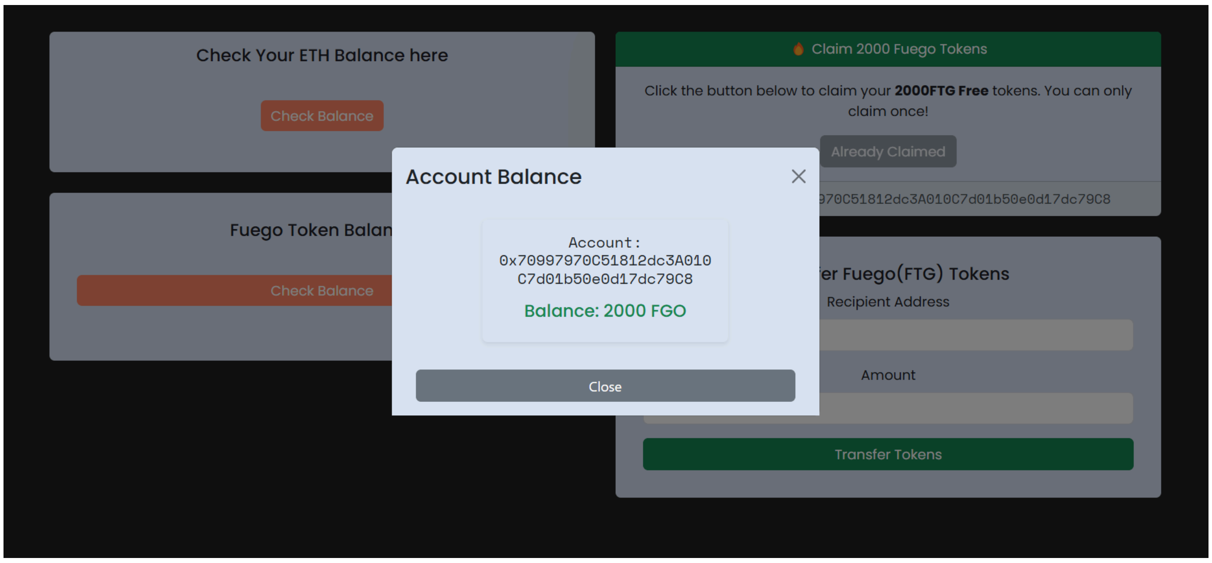


claim your **2000FTG Free** tokens. You can only claim once!

ETH Balance

```
10000.0 ETH
```

```
Connected: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
```



Click the button below to claim your **2000FTG Free** tokens. You can only claim once!

```
970C51812dc3A010C7d01b50e0d17dc79C8
```

```
Account:
0x70997970C51812dc3A010
C7d01b50e0d17dc79C8
```

Balance: 2000 FGO

- This function returns the **FGO balance** of the specified account.

Backend Deployment & Blockchain Interaction

The contract is deployed to an Ethereum blockchain (either **local Hardhat**, **Sepolia Testnet**, or **Mainnet**). The DApp interacts with this contract via **ethers.js**.

Contract Deployment Process

1. The smart contract is compiled and deployed using **Hardhat**.
2. Once deployed, the **contract address** is saved and used in the frontend.

How the Frontend Interacts with the Blockchain

- **ethers.js** is used to communicate with the smart contract.
- Functions like `mintOnce()`, `transferTokens()`, and `checkBalance()` are called from the contract.
- Transactions are signed and sent via **MetaMask**.

Error Handling & Security

- **Revert Messages**: The contract prevents double minting and unauthorized actions by reverting transactions.

- **Frontend Alerts**: Errors like insufficient funds, exceeded allowances, or contract failures are displayed to the user.

- **Gas Optimization**: The contract is optimized for minimal gas fees.

Conclusion

The **Fuego DApp** provides an easy and secure way to mint, transfer, and check token balances. Users only get **one free mint**, ensuring fairness, while transfers and balance checks work seamlessly via **ethers.js**.