Vault Test Contract

Project Overview

Project Overview

Vault smart contract is the one that users deposit and withdraw the whitelisted ERC20 token.

The contract have been created for the test purpose.

It allows admin add or remove ERC20 tokens and another admins.

It also has the pause and unpause functions for the unexpected accident to secure the users' assets in the contract.

1. Functional Requirements

1.1. Roles

Vault project has two roles:

- **Admin**: Can add token to whitelist or remove from whitelist. Can add another admin or remove. Can pause or unpause the contract at any time.
- **User**: Can deposit or withdraw the whitelisted token.

1.2. Features

Vault project has the following features:

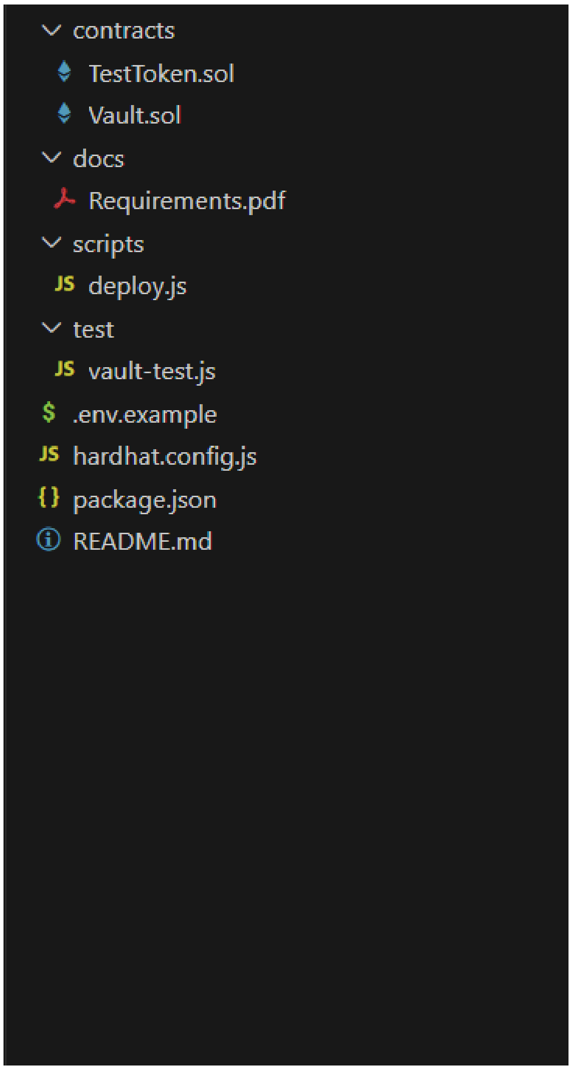
- Pause and unpause the contract. (Admin)
- Add or remove the token (whitelist). (Admin)
- Add or remove another admin. (Admin)
- Deposit whitelist token. (User)
- Withdraw whitelist token. (User)

2. Technical Requirements

This project has been developed with Solidity language, using Hardhat as a development environment. JavaScript was used for testing and deploying.

In additional, OpenZeppelin’s libraries are used in the project. All information about the contracts library and how to install it can be found in the Github.

In the project folder, the following structure is found:



```
contracts
    TestToken.sol
    Vault.sol
docs
    Requirements.pdf
scripts
    deploy.js
test
    vault-test.js
.env.example
hardhat.config.js
package.json
README.md
```

Start with README.md to find all basic information about the project structure and scripts that are required to test and deploy the contracts.

Inside the ./contracts folder, there Vault.sol file that explained in the section 2.2 of this document, and test token solidity file (for the testing).

In the ./test folder, there is vault-test.js file to provides the tests of the different methods of the main contracts using JavaScript.

The contracts can be deployed using the deploy.js script in ./script folder.

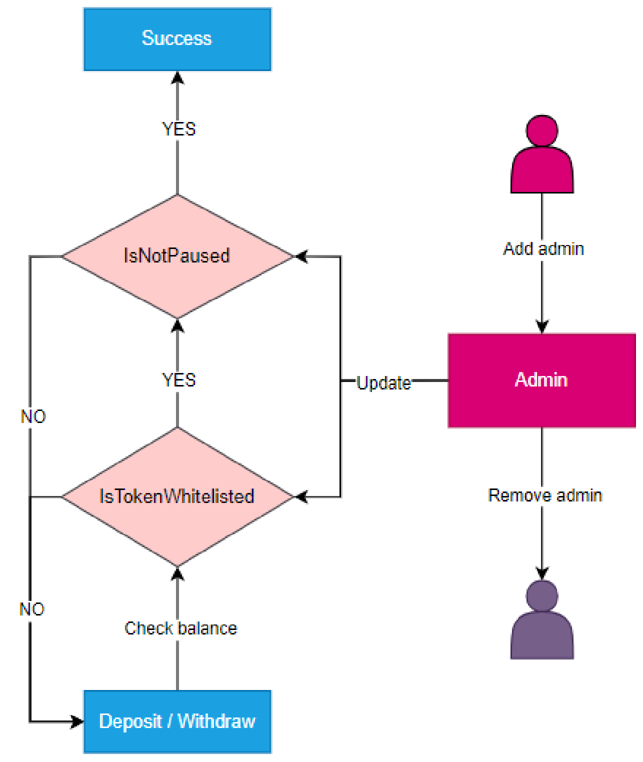
.env.example must be renamed to .env, and all required data should be provided.

The project configuration is found in hardhat.config.js, where dependencies are indicated.

Finally, this document can be found in ./docs.

2.1. Architecture Overview

The following diagram provides a general view of the Vault contracts structure and interactions between different functions.


Success
YES
IsNotPaused
YES
NO
IsTokenWhitelisted
NO
Check balance
Deposit / Withdraw
Update
Add admin
Admin
Remove admin

2.2. Contract Information

This section contains detailed information about the contracts used in the project.

2.2.1. Vault.sol

2.2.1.2. Events

There are following events:

- **Deposit**: Emitted when token deposited.
- **Withdrawal**: Emitted when token withdrew.
- **AdminAdded**: Emitted when new admin added.
- **AdminRemoved**: Emitted when admin removed.
- **TokenWhitelisted**: Emitted when new token added to whitelist
- **TokenRemovedFromWhiteList**: Emitted when token removed from whitelist

2.2.1.3. Modifiers

There are following modifiers:

- **onlyAdmins**: Check the caller is admin or not.
- **whenNotPaused**: Check contract is paused or not.

2.2.1.4. Functions

There are following functions:

- **constructor**: Add deployer to admin gruop.
- **deposit**: Deposit whitelisted token
- **withdraw**: Withdraw whitelisted token

- **pause**: Pause the contract. Only admin can call this function.
- **unpause**: Unpause the contract. Only admin can call this function.
- **whitelistToken**: Add token to whitelist. Only admin can call this function.
- **removeTokenFrmWhiltelist**: Remove token from whitelist. Only admin can call this function.
- **addAdmin**: Add new admin. Only admin can call this function.
- **removeAdmin**: Remove admin. Only admin can call this function.
- **isAdmin**: Check parameter(account) is admin or not. If admin return true, else return false.
- **isWhitelisted**: Check token is whitelist or not.