Soroban Escrow Smart contract

# 1. Overview

The Bounty Hunter Web Application is a platform where users can create and participate in bounties for various tasks.

The application will utilize the Stellar[1] blockchain using the Soroban[2] platform to create an Escrow Smart Contract that ensures secure fund management, authentication, and a review stage for accepting or rejecting work. The backend will be developed using Python Django, and the frontend will be built with React JS.

This project is for Escrow Smart contract on Soroban platform and is written in Rust[3] language.

# 2. Designing modules

We implemented this project with these modules – admin, fee, work, bounty, lib and test.

The following diagram depicts overall architecture.

creator

bounty

participant

work

applies

submits

approves/rejects

cancels/closes

Each module has the following responsibilities:

① admin

admin module is in charge of setting/getting administrator. It also supports checking administrator functionality.

② fee

fee module is in charge of setting/getting fee. It also supports checking fee functionality.

Fee information includes fee rate and wallet.

③ work

work module is in charge of submitting work. It supports creating, getting & writing of work.

④ bounty

bounty module is similar to real bounty. It supports creating, funding (put money in escrow), submitting work, approving & rejecting work, cancelling, closing work.

⑤ lib

lib module contains export functions so that web developers can use. These functions will be explained later.

⑥ test

test module contains test functions that we can do test with cargo. Currently it contains 4 test cases and they will be discussed later.

# 3. Data structures

Data structures are defined in storage\_types.rs. Constants, Enums and Structs are included.

① constants

Those include FEE\_DECIMALS, DEF\_FEE\_RATE, etc.

② Enums

Those include BountyStatus, WorkStatus, DataKey, etc. ErrorCodes are also included in here.

③ Structs

Those include FeeInfo, BountyInfo, etc.

# 4. Function descriptions

Lib module has the following functions:

- init

**Function**: Initialize contract.

**Parameters**: e is Environment variable, admin is the address of administrator.

- set\_admin

**Function**: Set administrator.

**Parameters**: e is Environment variable, admin is the address of administrator.

Only administrator can call this function.

- set\_fee

**Function**: Sets fee information.

**Parameters**: e is Environment variable, fee\_rate’s unit is 1/10FEE\_DECIMALS. fee\_wallet is the address of fee wallet.

- create\_bounty

**Function**: Creates a new bounty and funds reward to contract. Returns new bounty\_id on Success, errorcode on failure.

**Parameters**: creator is the address of bounty creator, name is bounty name, reward is payout amount, pay\_token is the address of token to use in Soroban (can be XLM(native) or other tokens), deadline is the end date (unit: 1s)

- apply\_bounty

**Function**: Worker applies for a bounty.

**Parameters**: participant is the address of worker and bounty\_id is the id of the bounty to apply.

- submit\_work

**Function**: Worker submits work for the bounty.

**Parameters**: participant is the address of worker, bounty\_id is the id of the bounty to submit, work\_repo is the URL of work repository.

- approve\_work

**Function**: Creator approves work for the bounty.

**Parameters**: creator is the address of bounty creator, work\_id is the id of the work submitted.

- reject\_work

**Function**: Creator approves work for the bounty.

**Parameters**: creator is the address of bounty creator, work\_id is the id of the work submitted.

- cancel\_bounty

**Function**: Creator cancels bounty.

**Parameters**: creator is the address of bounty creator, bounty\_id is id of the bounty to cancel.

- close\_bounty

**Function**: Closes expired bounty.

**Parameters**: creator is the address of bounty creator, bounty\_id is id of the bounty to close.

The following functions are accessory:

- token\_balances

Returns balance of token for the specified account.

# 5. Test cases

- TestCase1: Approve work

Sets fee, creates bounty, applies to bounty, submits work, approves work.

Here, check authentication and balance transfers after creating bounty and approving work.

- TestCase2: Reject work

Sets fee, creates bounty, applies to bounty, submits work, rejects work.

Here, check authentication and balance transfers after creating bounty and rejecting work.

- TestCase3: Cancel bounty

Sets fee, creates bounty, cancels bounty.

Here, check authentication and balance transfers after creating bounty and cancelling bounty.

- TestCase4: Close bounty

Sets fee, creates bounty, closes bounty.

Here, check authentication and balance transfers after creating bounty and closing bounty.

# 6. Build, test and deploy

① PreInstallation

* curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
* rustup target add wasm32-unknown-unknown
* cargo install --version 20.0.0-rc1 soroban-cli
* sudo apt install build-essential
* soroban

② Tests

* cd path\_of\_project
* make test

③ Deployments

* make build
* soroban contract deploy \

--wasm target/wasm32-unknown-unknown/release/[project-name].wasm \

--source <admin> \

--rpc-url https://rpc-futurenet.stellar.org:443 \

--network-passphrase 'Test SDF Future Network ; October 2022'

A new contract id is generated on success.

# References

[1] <https://developers.stellar.org/docs>

[2] <https://soroban.stellar.org>

[3] <https://doc.rust-lang.org/book/>

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Who** | **Contents** | **Note** |
| 2023/09/02 | Ivan Lesoviy | Draft version |  |
| 2023/09/05 | Ivan Lesoviy | Fixed soroban-cli part |  |
| 2023/10/08 | Ivan Lesoviy | Upgraded for Preview 11 |  |