**DESCRIPTION OF SMART CONTRACT OF TRILOBYTE METHODS**

1. `**vault\_req**`:

- This function is used to create a new vault request.

- It takes various parameters, including the invoker's address, loan information, lent token address, deployment configurations, payment actors, lender details, permission flag, debt token names, and repayment bytes.

- It returns a u128 value.

2. `**deposit**`:

- Investors use this method to fund the smart contract account according to the vault configuration. All the amount is added into the lending pool, and investors receive debt tokens equivalent to their deposit amount.

- Parameters include the invoker's address, the funding amount, fees, and a unique ID.

3. `**wdraw\_pool**`:

- Investors can withdraw funds from the lending pool if the phase is in the funding stage.

- It requires the invoker's address, the withdrawal amount, and a unique ID.

4. `**approval**`:

- Borrowers use this method to approve the withdrawal of a loan amount.

- It takes the borrower's address and a unique ID.

5. `**borrow**`:

- Borrowers can borrow an amount from the lending pool using this method.

- It requires the borrower's address, the loan amount, and a unique ID.

6. `**repay**`:

- Borrowers use this method to repay the lent tokens to the smart contract account, and the repayment amount is added into the cash pool.

- It takes the borrower's address, the repayment amount, and a unique ID.

7. `**wdraw\_from\_cashpool**`:

- Borrowers can withdraw lent tokens from the cash pool if there are available funds. In exchange, investors receive debt tokens, and borrowers get lent tokens with interest.

- Parameters include the borrower's address, the withdrawal amount, and a unique ID.

8. `**withdraw**`:

- Lenders with debt tokens can withdraw their lent tokens from the smart contract account if there is any amount in the EMI pool.

- It takes environmental data, the lender's address, the withdrawal amount, and a unique ID.

9. `**renegotiation**`:

- Borrowers can update the vault's state, but it requires the borrower and all investors to sign the update.

- Parameters include environmental data, loan information, a flag, payment actors, lender details, permission flag, repayment bytes, and a unique ID.

10. `**vault**`:

- This function retrieves the state of a vault based on a unique ID (soroban id) and returns a `Vault` data structure.

11. `**set\_admin**`:

- This method is used to set the admin for the smart contract.

- It takes, the current administrator's address, and the new admin's address.

- if you set admin, then if you want to set the admin again you need to sign this method with the previous admin,

- first, we have to set admin before calling any method

12. `**get\_admin**`:

- It retrieves the admin's address.

- It takes environmental data and the admin's address as a parameter and returns an `Address`.

14. `**update\_payments**`:

- This function updates the next payment date and transfers EMI from the cash pool to the EMI pool if the EMI due date has arrived.

15. `**get\_vault\_id**`:

- Retrieves the vault ID based on environmental data and returns a u128 value.

16. `**calculate\_payments**`:

- It calculates various payment-related values such as pmt, ipmt, and ppmt, which are calculated from JavaScript.

- Parameters include payment values, current principal, and a unique ID.

**How to deploy & run smart contract**

**Prerequisites:-**

**>Install Rust:-**

**curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh**

**>Install the target:-**

**rustup target add wasm32-unknown-unknown**

**>Install the Soroban CLI:-**

**cargo install --locked --version 20.0.0-rc.4.1 soroban-cli**

**>This is the doc for setup**

[**https://soroban.stellar.org/docs/getting-started/setup**](https://soroban.stellar.org/docs/getting-started/setup)

**To run and deploy smart contract**

**>GIT HUB LINK**

<https://github.com/orunpay/project-orun/tree/development>

Latest code branch name :- development

**Steps**

1. **clone repo from git,**
2. **cd smartcontract**
3. **CONTRACT BUILD**

cargo build --target wasm32-unknown-unknown --release

1. **if you want to run test case use :**

“cargo test --package trilobyte --lib -- test --nocapture --test-threads=1”

**5 Run deploy command:**

**>DEPLOY CONTRACT ON FUTURENET**

soroban contract deploy --wasm target/wasm32-unknown-unknown/release/trilobyte.wasm --source PrivateKey --rpc-url https://rpc-futurenet.stellar.org:443 --network-passphrase 'Test SDF Future Network ; October 2022'

**>DEPLOY CONTRACT ON TESTNET**

**In the below command, we will use the private key of anyone in place of hello**

soroban contract deploy --wasm target/wasm32-unknown-unknown/release/trilobyte.wasm --source **SCFET5HGH7QPBJKFF3EMDQMJQNYQNXQ5M6WEPAQ6A2FCLWAFNHLROZNP** --rpc-url https://soroban-testnet.stellar.org --network-passphrase 'Test SDF Network ; September 2015'

You will get a contract id save that id

**All the Address:-**

| **Accounts** | **Seed Phrase** | **Public Key** | **Secret Key** |
| --- | --- | --- | --- |
| ***Issuer*** |  | **GBO527FRRCR3STZ456CNBI6PJFOFMCINNI46T7F2NI7YHYTR6G3II5ZS** | **SCFET5HGH7QPBJKFF3EMDQMJQNYQNXQ5M6WEPAQ6A2FCLWAFNHLROZNP** |
| **Investor 1** | **1-bundle 2-hint 3-chef 4-room 5-problem 6-broccoli 7-used 8-supply 9-duck 10-elegant 11-lucky 12-kiwi** | **GBKAYNS4Q77QXKCPIGOYJANGHJD3ISVYW7OIJ7X436TG6BRWOF2VNIUS** | **SCRUBXFCWRQBROBRO6KQZICY5WTZNCTKTIUS7POCMDVF4UAHE6Y6OBSG** |
| **Investor 2** | **1-problem 2-car 3-parent 4-industry 5-genre 6-fresh 7-decide 8-fringe 9-text 10-father 11-target 12-option** | **GCBJ77UJOCUTTWS7F7DGHIQCQPPCPIP3UBSDXBXAUEKTNDWPVRHVXHOW** | **SB3DUZEFDK7QEIKJCCFTST75ZTR5LY4QSGGOCJR2H3FIWKA6CF7KPQ63** |
| **Investor 3** | **1-blade 2-escape 3-tool 4-danger 5-tower 6-flash 7-expand 8-orange 9-hamster 10-kit 11-frame 12-parade** | **GC2K2VWCT7WG4ZR7J3SYWCRPP5M26ZBFLU4KFZUEQVWOGCCG46CLHOSP** | **SAZH5MM2RAJZHMDPZYBKELWYNSBZ6WLDJV3ZUQTX6447LEQ7DFX5SFKX** |
| **Borrower** | **1-gasp 2-remember 3- maple 4- access 5- exhibit 6- visa 7- kitchen 8- hybrid 9- busy 10- search 11- matter 12- mix** | **GDNX3F6UZ32K2VWEIG7VBF27XFMV7CRWK6UAG4ESH6OYGZTQRG77UP5R** | **SBI64OPGA3RINC2XZQSAL2JWTHJQID7KYWNIBX4UY3UVLHMQKY6DIXSK** |
| **Payment Actor 1** | **1-illness 2-train 3-portion 4-soldier 5-increase 6-advance 7-federal 8-dolphin 9-fish 10-evoke 11-eight 12-obey** | **GBBK36PUXK4K7WGYDZZMNWAUE62IV3MI6SJOOXDGZQARRSEU7ULBBCDD** | **SAATN5RGX7EUVKRY6BADV7WKOXMELQCC4GBNZK3IT57TJSUXANCP6A64** |

**How to create USDC Token and deploy to soroban network**

**Note : - there is two network stellar n/w and soroban n/w**

**https://laboratory.stellar.org/#account-creator?network=test**

**First create Token in stellar network :-**

1. **Fund all account :**

* Go to the Stellar Laboratory website,
* Go to create account tab,
* Past public key to input box and fund the account

1. **Buid**

* Go to build transaction tab
* Add borrower public key as a source,
* Select change Trust option
* Select Alphanumeric 4
* Add assert USDC
* Add issuer pub key
* Click sign in transaction signer

1. **Sign transaction**

* Add private key of borrower
* Click on submit and then submit the tractaction

**Fund account with USDC:**

1. **Build**
   1. Go to the build transaction tab
   2. Add issuer public key as a source,
   3. Select Payment option
   4. Add all the information
   5. Click sign in transaction signer
2. **Sign transaction** 
   1. Add the private key of the issuer
   2. Click on submit and then submit the tractaction

**Note : thats how you create a token on stellar network now you have to deploy it to the soroban network using this command:**

**>DEPLOY CUSTOM TOKEN ON FUTURNET**

soroban lab token wrap --asset="USDC:GBO527FRRCR3STZ456CNBI6PJFOFMCINNI46T7F2NI7YHYTR6G3II5ZS" --source SCFET5HGH7QPBJKFF3EMDQMJQNYQNXQ5M6WEPAQ6A2FCLWAFNHLROZNP --rpc-url https://rpc-futurenet.stellar.org:443/ --network-passphrase 'Test SDF Future Network ; October 2022'

**>DEPLOY CUSTOM TOKEN ON TESTNET**

soroban lab token wrap --asset="USDC:GBO527FRRCR3STZ456CNBI6PJFOFMCINNI46T7F2NI7YHYTR6G3II5ZS" --source SCFET5HGH7QPBJKFF3EMDQMJQNYQNXQ5M6WEPAQ6A2FCLWAFNHLROZNP --rpc-url https://soroban-testnet.stellar.org --network-passphrase 'Test SDF Network ; September 2015'

**>DEPLOY CUSTOM TOKEN ON TESTNET :-** we will get a NEXT\_PUBLIC\_TOKEN\_CONTRACT\_ID which we uses in our frontend env file as a NEXT\_PUBLIC\_TOKEN\_CONTRACT\_ID

**Invoke sc methods from Js client**

**>GITHUB LINK**

[https://github.com/divyarang26/Soroban\_js\_client](https://github.com/divyarang26/Soroban_js_client/tree/beta_new)

Latest code branch name :- testing

Use yarn to install the dependency

**>RUN COMMAND**

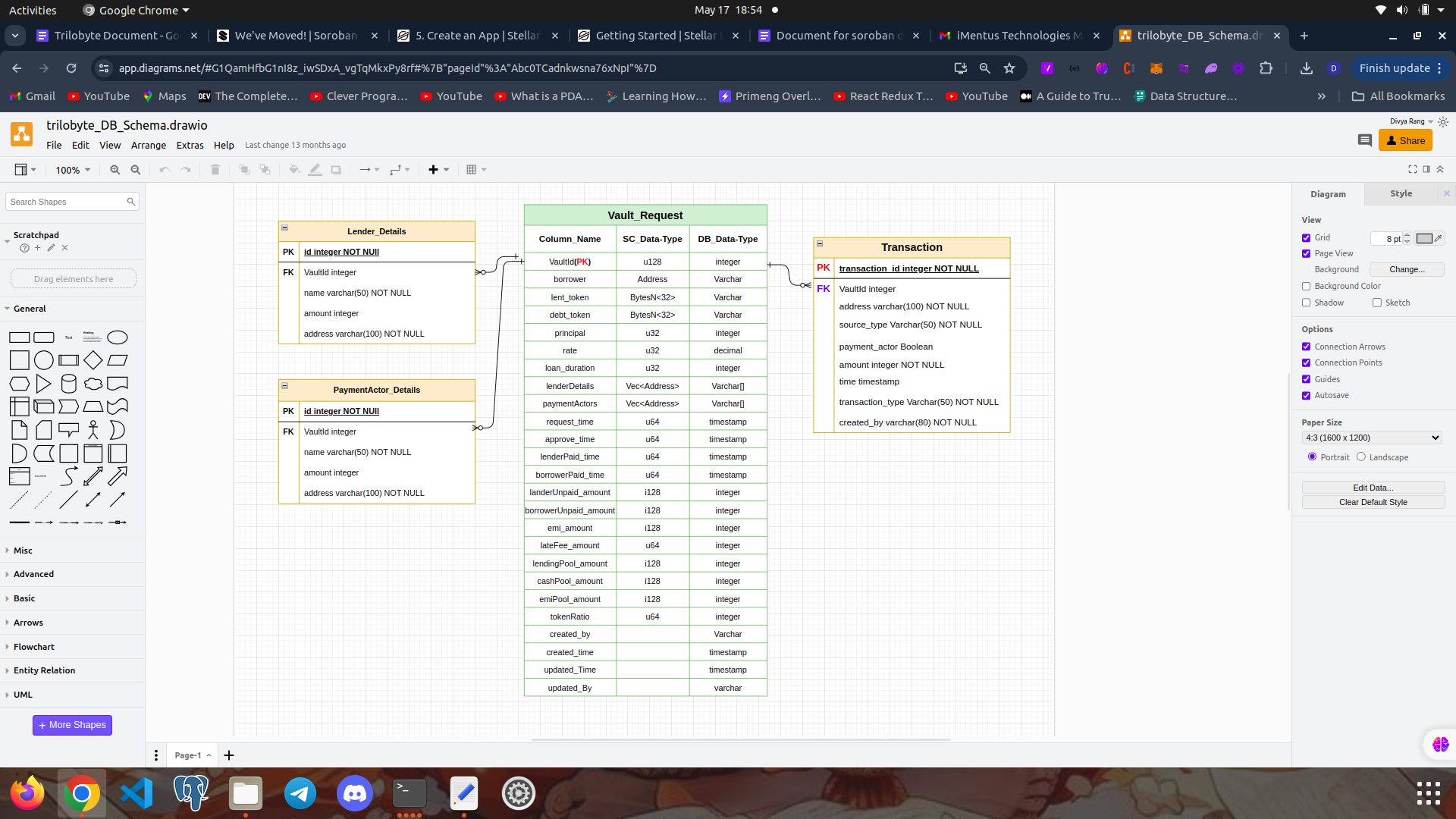
yarn ts-node src/FILENAME

**note:- If you deploy a new contract we have to set admin by invoke set\_admin method of smart contract just run this command yarn ts-node src/2\_set\_admin if you did not set admin other methods will not work**

Flow diagram

<https://www.figma.com/board/1v7cBkdL5rZZk7h5vDZL2G/Trilobyte?node-id=0-1&t=hnYupA8njJ2fyNNs-0>

Database schema



**How to update token wasm file**

**Clone this repo:-**

<https://github.com/stellar/soroban-examples>

**Go to token contract:-**

1. Cd token
2. Build contract using build command,
3. Then copy the wasm file from target/wasm32-unknown-unknown/release/soroban\_token\_contract.wasm
4. Then paste it to the token folder of our contract

**How to update smart contract version**

Go to <https://developers.stellar.org/docs/reference/software-versions> this site and check the version od sorban rust sdk

1. Then update cargo.toml
2. Delete cargo.lock and target file
3. Run build command

HOw to insatall wasm hash

soroban contract install --wasm token/soroban\_token\_contract.wasm --source SBGOWRNNOYK2RQ277HGGEGW7TYS7NWCE65GH76S33LZRQ7V5N4BOD4LD --rpc-url https://soroban-testnet.stellar.org --network-passphrase 'Test SDF Network ; September 2015'

**What is token client**

When evert you want to invoke token contract methods you have to create client on smart contract

**Update vault flow**

1. [**https://trilobyte-ui.herokuapp.com/vault-details/88/updateVault/**](https://trilobyte-ui.herokuapp.com/vault-details/88/updateVault/) **fill all the data invoke update method by borrower addres**
2. [**https://trilobyte-ui.herokuapp.com/vault-details/88/sign/**](https://trilobyte-ui.herokuapp.com/vault-details/88/sign/) **sign the transaction by all the investor by click sign button once all the investor sign the traction**
3. **Refresh the page**
4. **Click approve button sign by borrower**

**Urls**

[**https://trilobyte-ui.herokuapp.com/**](https://trilobyte-ui.herokuapp.com/)

[**https://app-futurenet.trilobyte.finance/**](https://app-futurenet.trilobyte.finance/)

[**https://test-api.trilobyte.finance/**](https://test-api.trilobyte.finance/)

[**https://trilobyte-api.herokuapp.com/**](https://trilobyte-api.herokuapp.com/)

**https://token-playground.gitbook.io/guide/index/6\_wrap\_a\_token\_from\_classic\_to\_soroban**