0.7

Tetu Finance Process Quality Review

Overview

This is a Tetu.io Process Quality Review completed on 01/11/2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nick of DeFiSafety. Check out our Telegram.

The final score of the review is **87%,** a **PASS.**The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is **70%.**

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

Chain

This section indicates the blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

✓ **Chain:** Polygon

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum
Solana

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the following questions:

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

✓ **Answer:** 100%

They are available at website https://docs.tetu.io/tetu-io/protocol/addresses, as indicated in the Appendix.

Guidance:

100%	Clearly labelled and on website, docs or repo, quick to find
70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Addresses in mainnet.json, in discord or sub graph, etc
20%	Address found but labeling not clear or easy to find
0%	Executing addresses could not be found

2) Is the code actively being used? (%)

Answer: 100%

Activity is 100+ transactions a day on contract *Controller*, as indicated in the Appendix.

Guidance:

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

3) Is there a public software repository? (Y/N)

Answer: Yes

GitHub: https://github.com/tetu-io

Is there a public software repository with the code at a minimum, but also normally test and scripts. Even if the repository was created just to hold the files and has just 1 transaction, it gets a **"Yes"**. For teams with private repositories, this answer is **"No"**.

4) Is there a development history visible? (%)

Answer: 100%

At 14 branches and 253 commits, this protocol has an impressive development history given it's relative youth.

This metric checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100%	Any one of 100+ commits, 10+branches
70%	Any one of 70+ commits, 7+branches
50%	Any one of 50+ commits, 5+branches
30%	Any one of 30+ commits, 3+branches
0%	Less than 2 branches or less than 30 commits

5) Is the team public (not anonymous)? (Y/N)

> ✓ **Answer:** Yes

Location: https://github.com/tetu-io/tetu-contracts/graphs/contributors

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, then this question is a **"No"**.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

6) Is there a whitepaper? (Y/N)
7) Are the basic software functions documented? (Y/N)
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
10) Is it possible to trace from software documentation to the implementation in
code (%)

6) Is there a whitepaper? (Y/N)

> ✓ **Answer:** Yes

Location: https://docs.tetu.io/tetu-io/

7) Are the basic software functions documented? (Y/N)

> ✓ **Answer:** Yes

The docs contain limited software function documentation.

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

Answer: 50%

Some software function documentation was found, though governance and the DEX (at least) are not covered.

Guidance:

100%	All contracts and functions documented
80%	Only the major functions documented
79-1%	Estimate of the level of software documentation
0%	No software documentation

How to improve this score:

This score can be improved by adding content to the software functions document such that it comprehensively covers the requirements. For guidance, refer to the SecurEth System Description Document. Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)

Answer: 38%

Code examples are in the Appendix. As per the SLOC, there is 38% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Guidance:

100%	CtC > 100 Useful comments consistently on all code
90-70%	CtC > 70 Useful comment on most code
60-20%	CtC > 20 Some useful commenting
0%	CtC < 20 No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the SecurEth Software Requirements.

10) Is it possible to trace from software documentation to the implementation in code (%)

Answer: 60%

Most functions are identified and there is association between the code and the documents, though there is no traceability.

Guidance:

100%	Clear explicit traceability between code and documentation at a requirement level for all code
60%	Clear association between code and documents via non explicit traceability
40%	Documentation lists all the functions and describes their functions
0%	No connection between documentation and code

How to improve this score:

This score can improve by adding traceability from documentation to code such that it is clear where each outlined function is coded in the source code. For reference, check the SecurEth guidelines on traceability.

Testing

This section looks at the software testing available. It is explained in this document. This section answers the following questions;

11) Full test suite (Covers all the deployed code) (%)
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
13) Scripts and instructions to run the tests (Y/N)
14) Report of the results (%)
15) Formal Verification test done (%)
16) Stress Testing environment (%)

11) Is there a Full test suite? (%)

✓ **Answer:** 100%

Code examples are in the Appendix. As per the SLOC, there is 136% testing to code (TtC).

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%. However the reviewers best judgement is the final deciding factor.

Guidance:

100%	TtC > 120% Both unit and system test visible
80%	TtC > 80% Both unit and system test visible
40%	TtC < 80% Some tests visible
0%	No tests obvious

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

Answer: 88%

Tetu's code is 88% covered.

Guidance:

100%	Documented full coverage
99-51%	Value of test coverage from documented results
50%	No indication of code coverage but clearly there is a reasonably complete set of tests
30%	Some tests evident but not complete
0%	No test for coverage seen

How to improve this score:

This score can improved by adding tests that achieve full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

Answer: Yes

Scripts with instructions are found in the GitHub.

14) Report of the results (%)

Answer: 70%

A detailed test report is found in the GitHub.

Guidance:

100%	Detailed test report as described below
70%	GitHub code coverage report visible
0%	No test report evident

15) Formal Verification test done (%)

Answer: 0%

Tetu has not undertaken formal verification testing yet.

16) Stress Testing environment (%)

✓ **Answer:** 100%

Tetu has been deployed to Rinkeby.

Security

This section looks at the 3rd party software audits done. It is explained in this document. This section answers the following questions;

17) Did 3rd Party audits take place? (%)
18) Is the bounty value acceptably high?

17) Did 3rd Party audits take place? (%)

✓ **Answer:** 100%

Mutliple audits have taken place on Tetu, both before and after launch. However, the DefiYield audit was started before deployment.

Guidance:

100%	Multiple Audits performed before deployment and results public and implemented or not required
90%	Single audit performed before deployment and results public and implemented or not required
70%	Audit(s) performed after deployment and no changes required. Audit report is public
50%	Audit(s) performed after deployment and changes needed but not implemented
20%	No audit performed
0%	Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, (where question 1 is 0%)

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

18) Is the bounty value acceptably high (%)

⚠ **Answer:** 20%

Tetu has an active program offering $29k.

Guidance:

100%	Bounty is 10% TVL or at least $1M AND active program (see below)
90%	Bounty is 5% TVL or at least 500k AND active program
80%	Bounty is 5% TVL or at least 500k
70%	Bounty is 100k or over AND active program
60%	Bounty is 100k or over
50%	Bounty is 50k or over AND active program
40%	Bounty is 50k or over
20%	Bug bounty program bounty is less than 50k
0%	No bug bounty program offered

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this document. The questions this section asks are as follow;

19) Can a user clearly and quickly find the status of the admin controls?
20) Is the information clear and complete?
21) Is the information in non-technical terms that pertain to the investments?
22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the access controls (%)

✓ **Answer:** 100%

Tetu's access control information is easy to find.

Guidance:

100%	Clearly labelled and on website, docs or repo, quick to find
70%	Clearly labelled and on website, docs or repo but takes a bit of looking
40%	Access control docs in multiple places and not well labelled
20%	Access control docs in multiple places and not labelled
0%	Admin Control information could not be found

20) Is the information clear and complete (%)

✓ **Answer:** 100%

Tetu's contracts are immutable.

Guidance:

All the contracts are immutable -- 100% OR

a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
c) The capabilities for change in the contracts are described -- 30%

How to improve this score:

Create a document that covers the items described above. An example is enclosed.

21) Is the information in non-technical terms that pertain to the investments (%)

✓ **Answer:** 100%

Contracts are immutable.

Guidance:

100%	All the contracts are immutable
90%	Description relates to investments safety and updates in clear, complete non-software l language
30%	Description all in software specific language
0%	No admin control information could not be found

22) Is there Pause Control documentation including records of tests (%)

✓ **Answer:** 80%

Tetu explains its pause function well, but does not document a test.

Guidance:

100%	All the contracts are immutable or no pause control needed and this is explained OR
100%	Pause control(s) are clearly documented and there is records of at least one test within 3 months
80%	Pause control(s) explained clearly but no evidence of regular tests
40%	Pause controls mentioned with no detail on capability or tests
0%	Pause control not documented or explained

How to improve this score**:**

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

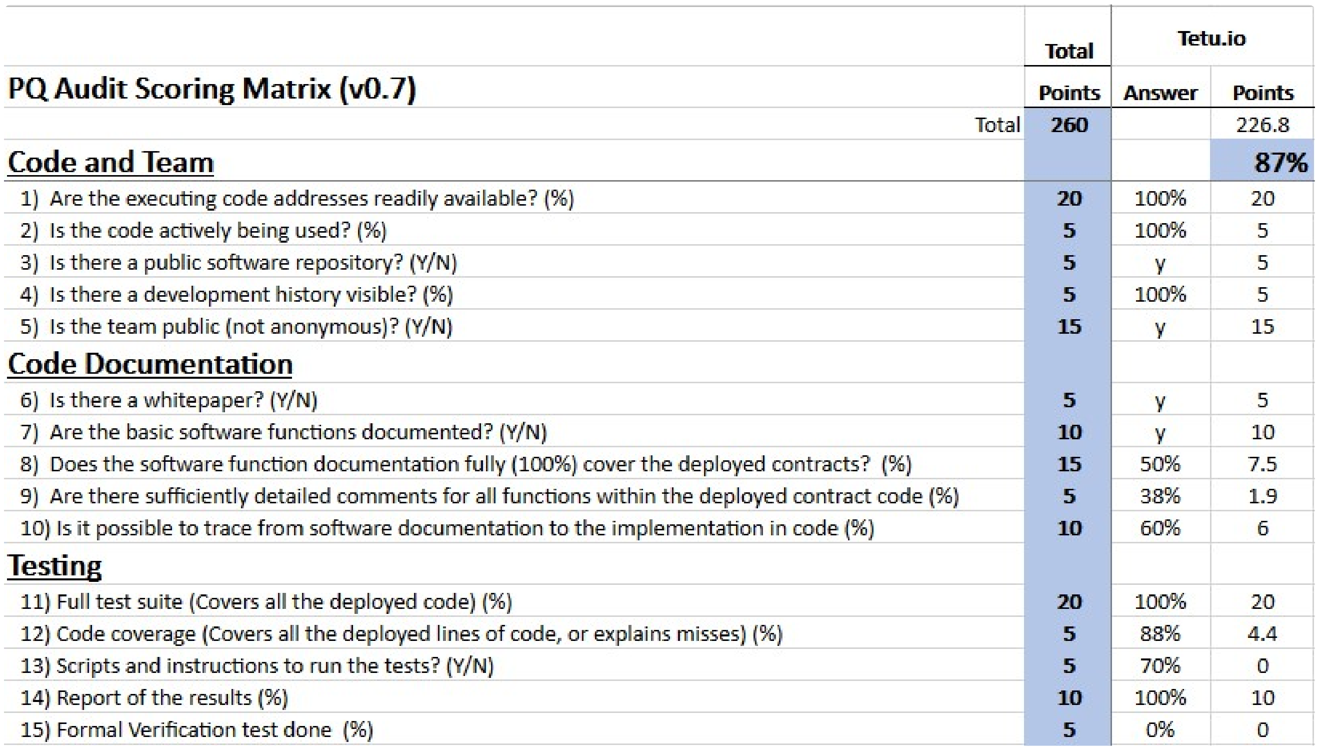
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got EthFoundation funding to assist in their development.

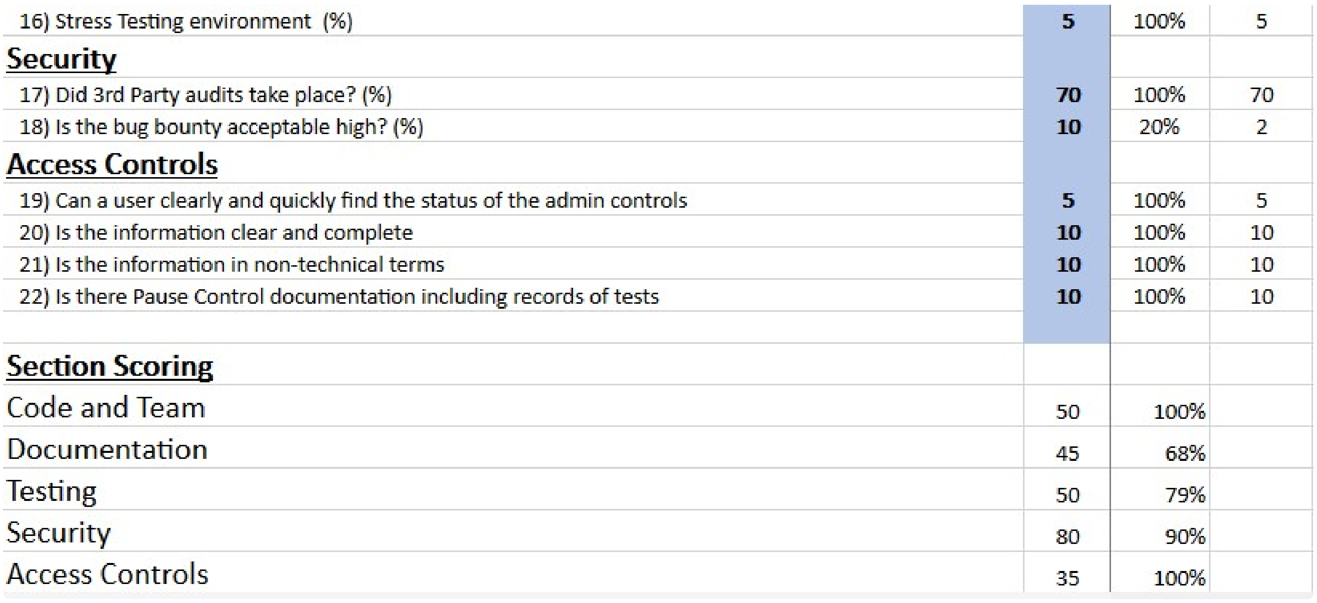
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix



	Total	Tetu.io	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		226.8
Code and Team			**87%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	y	5
4) Is there a development history visible? (%)	**5**	100%	5
5) Is the team public (not anonymous)? (Y/N)	**15**	y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	y	5
7) Are the basic software functions documented? (Y/N)	**10**	y	10
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	**15**	50%	7.5
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	38%	1.9
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	88%	4.4
13) Scripts and instructions to run the tests? (Y/N)	**5**	70%	0
14) Report of the results (%)	**10**	100%	10
15) Formal Verification test done (%)	**5**	0%	0

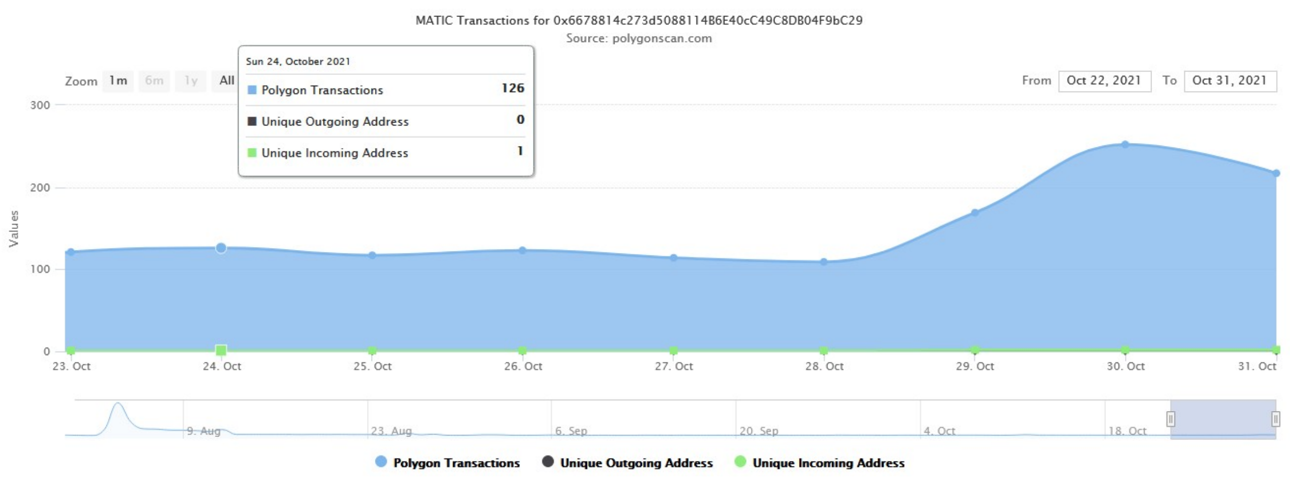


16) Stress Testing environment (%)	**5**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	20%	2
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	100%	5
20) Is the information clear and complete	**10**	100%	10
21) Is the information in non-technical terms	**10**	100%	10
22) Is there Pause Control documentation including records of tests	**10**	100%	10
Section Scoring			
Code and Team	50	100%	
Documentation	45	68%	
Testing	50	79%	
Security	80	90%	
Access Controls	35	100%	

Executing Code Appendix

Name	Address
TETU Token	0x255707B70BF90aa112006E1b07B9AeA6De021424
xTETU Profit Share Token	0x225084d30cc297f3b177d9f93f5c3ab8fb6a1454
Controller	0x6678814c273d5088114B6E40cC49C8DB04F9bC29
Announcer	0x286c02C93f3CF48BB759A93756779A1C78bCF833
FeeRewardForwarder	0xd055b086180cB6dac888792C9307970Ed10CF137
Bookkeeper	0x0A0846c978a56D6ea9D2602eeb8f977B21F3207F
NotifyHelper - 1.1.0-r1	0x560471ab39C3Eb26D63aB3b2A5b9835764C998ea

Code Used Appendix



Example Code Appendix

```
pragma solidity 0.8.4;

import "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol";
import "@openzeppelin/contracts/utils/Address.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
import "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol";
import "@openzeppelin/contracts/utils/math/Math.sol";
import "../../third_party/uniswap/IUniswapV2Pair.sol";
import "../../third_party/uniswap/IUniswapV2Router02.sol";
import "../price/IPriceCalculator.sol";
import "./PayrollClerkStorage.sol";

/// @title Disperse salary to workers
/// @author belbix
contract PayrollClerk is PayrollClerkStorage {
  using SafeMath for uint256;
  using SafeERC20 for IERC20;

  event WorkerRateUpdated(address indexed worker, uint256 value);
  event WorkerNameUpdated(address indexed worker, string value);
  event WorkerRoleUpdated(address indexed worker, string value);
  event TokenChanged(address[] tokens, uint256[] ratios);
  event SalaryPaid(address indexed worker, uint256 usdAmount, uint256 workedHours, uint256
  event TokenMoved(address token, uint256 amount);

  function initialize(address _controller, address _calculator) external initializer {
    require(_calculator != address(0), "zero calculator address");
    Controllable.initializeControllable(_controller);

    bytes32 slot = _CALCULATOR_SLOT;
    assembly {
```

```
    sstore(slot, _calculator)
  }
}

function allWorkers() external view returns (address[] memory) {
  return workers;
}

function workersLength() external view returns (uint256) {
  return workers.length;
}

function multiplePay(address[] calldata _workers, uint256[] calldata _workedHours)
external onlyControllerOrGovernance {
  require(_workers.length == _workedHours.length, "wrong arrays");
  for (uint256 i = 0; i < _workers.length; i++) {
    pay(_workers[i], _workedHours[i]);
  }
}

function pay(address worker, uint256 _workedHours) public onlyControllerOrGovernance {
  require(baseHourlyRates[worker] != 0, "worker not registered");

  uint256 totalSalaryUsd;
  for (uint256 i = 0; i < tokens.length; i++) {
    (uint256 salaryUsd, uint256 salaryToken) = computeSalary(worker, _workedHours, tokens
    require(salaryToken <= ERC20(tokens[i]).balanceOf(address(this)), "not enough fund")
    IERC20(tokens[i]).safeTransfer(worker, salaryToken);
    totalSalaryUsd = totalSalaryUsd.add(salaryUsd);
  }
  workedHours[worker] = workedHours[worker].add(_workedHours);
  earned[worker] = earned[worker].add(totalSalaryUsd);
  emit SalaryPaid(worker, totalSalaryUsd, _workedHours, hourlyRate(worker));
}

function computeSalary(address worker, uint256 _workedHours, address token)
public view returns (uint256 salaryUsd, uint256 salaryToken) {
  uint256 tPrice = IPriceCalculator(calculator()).getPriceWithDefaultOutput(token);
  uint256 hRate = hourlyRate(worker);
  salaryUsd = hRate.mul(_workedHours).mul(1e18)
  .mul(tokenRatios[token]).div(FULL_RATIO);

  // return token amount with token decimals
  salaryToken = salaryUsd.mul(1e18).div(tPrice)
  .mul(10 ** ERC20(token).decimals()).div(1e18);
}

function hourlyRate(address worker) public view returns (uint256) {
  uint256 ratio = 1;
  if (boostActivated[worker]) {
    ratio = Math.min(workedHours[worker].div(BUST_STEP).add(1), MAX_BOOST);
  }
  return Math.min(baseHourlyRates[worker].mul(ratio), MAX_HOURLY_RATE);
```

```
}

/// if a wallet changed we need a way to migration
function changeWorkerAddress(address oldWallet, address newWallet) external onlyControll
  uint256 idx = workerIndex(oldWallet);
  require(idx != type(uint256).max, "worker not registered");

  workerNames[newWallet] = workerNames[oldWallet];
  workerNames[oldWallet] = "";

  workerRoles[newWallet] = workerRoles[oldWallet];
  workerRoles[oldWallet] = "";

  baseHourlyRates[newWallet] = baseHourlyRates[oldWallet];
  baseHourlyRates[oldWallet] = 0;

  workedHours[newWallet] = workedHours[oldWallet];
  workedHours[oldWallet] = 0;

  earned[newWallet] = earned[oldWallet];
  earned[oldWallet] = 0;

  boostActivated[newWallet] = boostActivated[oldWallet];
  boostActivated[oldWallet] = false;

  workers[idx] = newWallet;
}

function addWorkers(
  address[] calldata _workers,
  uint256[] calldata rates,
  string[] calldata names,
  string[] calldata roles,
  bool[] calldata boosts
)
external onlyControllerOrGovernance {
  require(
    _workers.length == rates.length
    && _workers.length == names.length
    && _workers.length == roles.length
    && _workers.length == boosts.length
  , "wrong arrays");
  for (uint256 i = 0; i < _workers.length; i++) {
    addWorker(_workers[i], rates[i], names[i], roles[i], boosts[i]);
  }
}

function addWorker(
  address worker,
  uint256 rate,
  string calldata name,
  string calldata role,
  bool boost
```

```
    bool boost
  ) public onlyControllerOrGovernance {
    require(baseHourlyRates[worker] == 0, "worker already registered");
    workers.push(worker);
    setWorkerName(worker, name);
    setWorkerRole(worker, role);
    setBaseHourlyRate(worker, rate);
    boostActivated[worker] = boost;
  }

  function setWorkerName(address worker, string calldata name) public onlyControllerOrGove
    require(bytes(name).length != 0, "empty name");
    require(bytes(name).length < 20, "too big name");
    workerNames[worker] = name;
    emit WorkerNameUpdated(worker, name);
  }

  function setWorkerRole(address worker, string calldata role) public onlyControllerOrGove
    require(bytes(role).length != 0, "empty name");
    require(bytes(role).length < 20, "too big role");
    workerRoles[worker] = role;
    emit WorkerRoleUpdated(worker, role);
  }

  function setBaseHourlyRate(address worker, uint256 rate) public onlyControllerOrGovernan
    require(rate != 0, "zero rate");
    require(rate <= MAX_HOURLY_RATE, "too high rate");
    baseHourlyRates[worker] = rate;
    emit WorkerRateUpdated(worker, rate);
  }

  function changeTokens(address[] calldata _tokens, uint256[] calldata ratios)
  external onlyControllerOrGovernance {
    require(_tokens.length == ratios.length, "wrong arrays");
    tokens = _tokens;

    for (uint i = 0; i < _tokens.length; i++) {
      tokenRatios[_tokens[i]] = ratios[i];
    }

    checkTokenRatios();
    emit TokenChanged(_tokens, ratios);
  }

  function switchBoost(address worker, bool active) external onlyControllerOrGovernance {
    require(baseHourlyRates[worker] != 0, "worker not registered");
    boostActivated[worker] = active;
  }

  function checkTokenRatios() internal view {
    uint256 sum;
    for (uint256 i = 0; i < tokens.length; i++) {
      sum = sum.add(tokenRatios[tokens[i]]);
    }

    require(sum == FULL_RATIO, "invalid token ratios");
  }

  function workerIndex(address worker) public view returns (uint256){
    for (uint256 i = 0; i < workers.length; i++) {
      if (workers[i] == worker) {
        return i;
      }
    }
    return type(uint256).max;
  }

  /// @dev Move tokens to governance
  ///      This contract should contain only governance funds
  function moveTokensToGovernance(address _token, uint256 amount) external onlyControllerO
    uint256 tokenBalance = IERC20(_token).balanceOf(address(this));
    require(tokenBalance >= amount, "not enough balance");
    IERC20(_token).safeTransfer(IController(controller()).governance(), amount);
    emit TokenMoved(_token, amount);
  }

}
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	1	7708	1140	1806	4762	739

Comments to Code 1835/4770 = 38%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
TypeScript	38	8213	1537	182	6494	208

Tests to Code 6494/4770 = 136%