0.7

StakeWise Process Quality Review

Score: 79%

Overview

This is a StakeWise Process Quality Review completed on July 12th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is 79%, 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

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain: Ethereum**

Guidance:

Ethereum
Binance Smart Chain
Polygon

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: 20%

They are available at website https://github.com/stakewise/contracts/blob/master/networks/mainnet.md, as indicated in the Appendix.

Note: As you can see by the link provided above, the Mainnet contracts were not easy to find nor were they well-labelled.

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

How to improve this score:

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question towards the final score.

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

> Answer: 100%

Activity is 10 transactions a day on contract *Pool.sol*, 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/stakewise.

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%

With 396 commits and 6 branches, this is a robust software repository.

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://angel.co/company/stakewise/people.

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.stakewise.io/.

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

> ✓ Answer: Yes

Basic software functions (code) was found in https://docs.stakewise.io/smart-contracts.

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

> Answer: 50%

Almost all of the functions listed in StakeWise's documentation are only giving overall operation description, not software documentation.

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: 90%

Code examples are in the Appendix. As per the SLOC, there is 93% 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: 100%

There is clear and explicit traceability between software documentation and its implementation in code due to the fact that each software function outlined in StakeWise's GitBooks is a hyperlink to that same function's location in the source code within their GitHub.

Example: cancelDeposit (taken from the "Solos" section of their GitBooks).

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

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 287% 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: 99%

According to their codecov report, StakeWise has a code coverage of 99%.

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

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

Answer: Yes

Scrips/Instructions location: https://github.com/stakewise/contracts/tree/master/scripts, instructions can be found at the bottom of the "contracts" repository.

14) Report of the results (%)

Answer: 70%

Note: They would get 100% if they published their own test report in the corresponding repository. However, they still get a 70% for having that codecov report linked to their GitHub.

Guidance:

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

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

Answer: 0%

No evidence of a StakeWise Formal Verification test was found in their documentation, GitHub, or on the

web.

Note: They were audited by Runtime Verification, who do Formal Verifications, but did a Security Audit for StakeWise instead.

16) Stress Testing environment (%)

> ✓ Answer: 100%

There is evidence of StakeWise test-net usage at https://stakewise.medium.com/stakewise-the-1st-eth-staking-platform-to-join-topaz-testnet-33bd25687d74.

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%

Runtime Verification published a StakeWise security audit report on December 23rd 2020.

Certik published a StakeWise a first Staking security assessment report on April 18th 2021.

Certik published a StakeWise Staking security assessment report on June 1st 2021.

StakeWise was launched their mainnet on March 8th 2021, meaning they had 1 audit before launch. In addition, they launched their Staking contract on April 19th, while publishing an audit report from Certik the day before. Since StakeWise has had multiple audits pre-launch, they get 100% for this metric.

Note: A majority of the fix recommendations provided by Runtime and Certik were successfully implemented by the StakeWise team.

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, question

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: 0%

No evidence of a Stake Wise Bug Bounty program was found in their documentation or in web searches.

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%

Clearly labelled and found at https://docs.stakewise.io/governance/stakewise-dao.

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: 90%

a) DAO voting towards contract upgradeability is clearly outlined.

b) Gnosis safe multisig (4 out of 7 key holders needed) paired with a 51% community governance.

c) A broad overview of what contract parameters the DAO can vote to upgrade (or not) is described at https://docs.stakewise.io/governance/stakewise-dao.

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: 90%

All DAO governance and contract ownership information is detailed in clear, non-software-y language.

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

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.

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

> ⚠ Answer: 0%

No Pause Control or similar function was found in StakeWise's documentation, nor were tests found in their GitHub.

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.

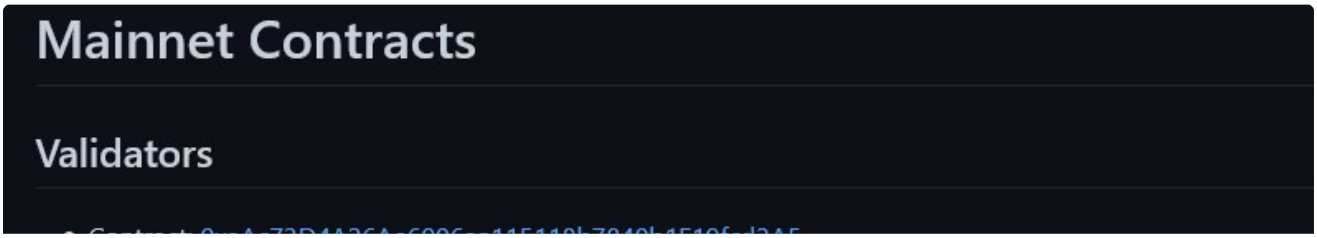
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

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

Scoring Appendix

	Total	StakeWise	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	260		204.6
Code and Team			**79%**
1) Are the executing code addresses readily available? (%)	20	20%	4
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	40%	6
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	5	93%	4.65
10) Is it possible to trace from software documentation to the implementation in code (%)	10	100%	10
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	99%	4.95
13) Scripts and instructions to run the tests? (Y/N)	5	Y	5
14) Report of the results (%)	10	70%	7
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	0%	0
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	90%	9
21) Is the information in non-technical terms	10	90%	9
22) Is there Pause Control documentation including records of tests	10	0%	0
Section Scoring			
Code and Team	50	68%	
Documentation	45	79%	
Testing	50	84%	
Security	80	88%	
Access Controls	35	66%	

Executing Code Appendix



- Contract: 0xaAc73D4A26Ae6906aa1151f8b7840b1F19fcd3A5
- Transaction: 0xdf4df05f116f81e505766f972142acc16479ade3961b7920ce5b434f4023f289

Pool

- Contract: 0xC874b064f465bdD6411D45734b56fac750Cda29A
- Transaction: 0xefa2f24402c9380a7721c725734809820ad8ac4f042f4e7f9aee77a1a54e662e

Upgrade to v1.1.0

- Implementation: 0xc8970E7C07c251625F9F93cE510b1D9c1a08d299
- Transaction: 0x32c727ebb5dbbc1bd89a3a21cf15d9e74dbc688b404b60a286a9ae858c4967e2

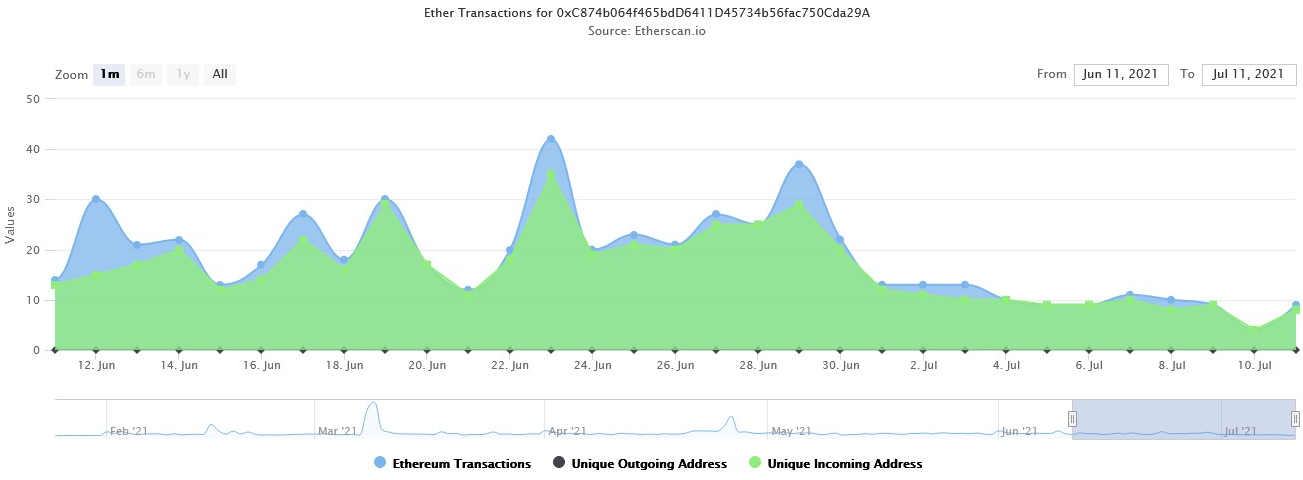
Pool Escrow

- Contract: 0x2296e122c1a20Fca3CAc3371357BdAd3be0dF079
- Transaction: 0xaf485028fa48fe546a72d0a26c8ea8adc6ae0c460faa3b5fab7513834c0e1155

Solos

- Contract: 0xEadCBA8BF9ACA93F627F31fB05470F5A0686CEca
- Transaction: 0xb06d6b7288424b04fab9136388c68cb9083e392bae00816c36e062dac3666dc2

Code Used Appendix



Example Code Appendix

```
/**
 * @title Pool
 *
 * @dev Pool contract accumulates deposits from the users, mints tokens and registers vali
 */
```

```
contract Pool is IPool, OwnablePausableUpgradeable {
    using SafeMathUpgradeable for uint256;

    // @dev Validator deposit amount.
    uint256 public constant VALIDATOR_DEPOSIT = 32 ether;

    // @dev Total activated validators.
    uint256 public override activatedValidators;

    // @dev Pool validator withdrawal credentials.
    bytes32 public override withdrawalCredentials;

    // @dev Address of the ETH2 Deposit Contract (deployed by Ethereum).
    IDepositContract public override validatorRegistration;

    // @dev Address of the StakedEthToken contract.
    IStakedEthToken private stakedEthToken;

    // @dev Address of the Validators contract.
    IValidators private validators;

    // @dev Address of the Oracles contract.
    address private oracles;

    // @dev Maps senders to the validator index that it will be activated in.
    mapping(address => mapping(uint256 => uint256)) public override activations;

    // @dev Total pending validators.
    uint256 public override pendingValidators;

    // @dev Amount of deposited ETH that is not considered for the activation period.
    uint256 public override minActivatingDeposit;

    // @dev Pending validators percent limit. If it's not exceeded tokens can be minted im
    uint256 public override pendingValidatorsLimit;

    /**
     * @dev See {IPool-upgrade}.
     * The `initialize` must be called before upgrading in previous implementation contrac
     * https://github.com/stakewise/contracts/blob/v1.0.0/contracts/collectors/Pool.sol#L4
     */
    function upgrade(
        address _oracles,
        uint256 _activatedValidators,
        uint256 _pendingValidators,
        uint256 _minActivatingDeposit,
        uint256 _pendingValidatorsLimit
    )
        external override onlyAdmin whenPaused
    {
        require(oracles == address(0), "Pool: already upgraded");
        oracles = _oracles;

```

```
59          pendingValidators = _pendingValidators;
60          activatedValidators = _activatedValidators;
61          emit ActivatedValidatorsUpdated(_activatedValidators, msg.sender);
62
63          minActivatingDeposit = _minActivatingDeposit;
64          emit MinActivatingDepositUpdated(_minActivatingDeposit, msg.sender);
65
66          pendingValidatorsLimit = _pendingValidatorsLimit;
67          emit PendingValidatorsLimitUpdated(_pendingValidatorsLimit, msg.sender);
68      }
69
70      /**
71       * @dev See {IPool-setWithdrawalCredentials}.
72       */
73      function setWithdrawalCredentials(bytes32 _withdrawalCredentials) external override on
74          withdrawalCredentials = _withdrawalCredentials;
75          emit WithdrawalCredentialsUpdated(_withdrawalCredentials);
76      }
77
78      /**
79       * @dev See {IPool-setMinActivatingDeposit}.
80       */
81      function setMinActivatingDeposit(uint256 _minActivatingDeposit) external override only
82          minActivatingDeposit = _minActivatingDeposit;
83          emit MinActivatingDepositUpdated(_minActivatingDeposit, msg.sender);
84      }
85
86      /**
87       * @dev See {IPool-setPendingValidatorsLimit}.
88       */
89      function setPendingValidatorsLimit(uint256 _pendingValidatorsLimit) external override
90          require(_pendingValidatorsLimit < 10000, "Pool: invalid limit");
91          pendingValidatorsLimit = _pendingValidatorsLimit;
92          emit PendingValidatorsLimitUpdated(_pendingValidatorsLimit, msg.sender);
93      }
94
95      /**
96       * @dev See {IPool-setActivatedValidators}.
97       */
98      function setActivatedValidators(uint256 newActivatedValidators) external override {
99          require(msg.sender == oracles || hasRole(DEFAULT_ADMIN_ROLE, msg.sender), "Pool: a
100
101         // subtract activated validators from pending validators
102         pendingValidators = pendingValidators.sub(newActivatedValidators.sub(activatedVali
103         activatedValidators = newActivatedValidators;
104         emit ActivatedValidatorsUpdated(newActivatedValidators, msg.sender);
105     }
106
107     /**
108      * @dev See {IPool-addDeposit}.
109      */
110     function addDeposit() external payable override whenNotPaused {
111         require(msg.value > 0, "Pool: invalid deposit amount");
```

```
112
113         // mint tokens for small deposits immediately
114         if (msg.value <= minActivatingDeposit) {
115             stakedEthToken.mint(msg.sender, msg.value);
116             return;
117         }
118
119         // mint tokens if current pending validators limit is not exceed
120         uint256 _pendingValidators = pendingValidators.add((address(this).balance).div(VAL
121         uint256 _activatedValidators = activatedValidators; // gas savings
122         uint256 validatorIndex = _activatedValidators.add(_pendingValidators);
123         if (validatorIndex.mul(1e4) <= _activatedValidators.mul(pendingValidatorsLimit.add
124             stakedEthToken.mint(msg.sender, msg.value);
125         } else {
126             // lock deposit amount until validator activated
127             activations[msg.sender][validatorIndex] = activations[msg.sender][validatorInd
128             emit ActivationScheduled(msg.sender, validatorIndex, msg.value);
129         }
130     }
131
132     /**
133      * @dev See {IPool-canActivate}.
134      */
135     function canActivate(uint256 _validatorIndex) external view override returns (bool) {
136         return _validatorIndex.mul(1e4) <= activatedValidators.mul(pendingValidatorsLimit.
137     }
138
139     /**
140      * @dev See {IPool-activate}.
141      */
142     function activate(address _account, uint256 _validatorIndex) external override whenNotP
143         require(_validatorIndex.mul(1e4) <= activatedValidators.mul(pendingValidatorsLimit
144
145         uint256 amount = activations[_account][_validatorIndex];
146         require(amount > 0, "Pool: invalid validator index");
147
148         delete activations[_account][_validatorIndex];
149         stakedEthToken.mint(_account, amount);
150         emit Activated(_account, _validatorIndex, amount, msg.sender);
151     }
152
153     /**
154      * @dev See {IPool-activateMultiple}.
155      */
156     function activateMultiple(address _account, uint256[] calldata _validatorIndexes) exte
157         uint256 toMint;
158         uint256 _activatedValidators = activatedValidators;
159         for (uint256 i = 0; i < _validatorIndexes.length; i++) {
160             uint256 validatorIndex = _validatorIndexes[i];
161             require(validatorIndex.mul(1e4) <= _activatedValidators.mul(pendingValidatorsL
162
163             uint256 amount = activations[_account][validatorIndex];
164             toMint = toMint.add(amount);
```

```
            toMint = toMint.add(amount);

            delete activations[_account][validatorIndex];

            emit Activated(_account, validatorIndex, amount, msg.sender);
        }
        require(toMint > 0, "Pool: invalid validator index");
        stakedEthToken.mint(_account, toMint);
    }

    /**
     * @dev See {IPool-registerValidator}.
     */
    function registerValidator(Validator calldata _validator) external override whenNotPaus
        require(validators.isOperator(msg.sender), "Pool: access denied");

        // register validator
        validators.register(keccak256(abi.encodePacked(_validator.publicKey)));
        emit ValidatorRegistered(_validator.publicKey, msg.sender);

        // update number of pending validators
        pendingValidators = pendingValidators.add(1);

        validatorRegistration.deposit{value : VALIDATOR_DEPOSIT}(
            _validator.publicKey,
            abi.encodePacked(withdrawalCredentials),
            _validator.signature,
            _validator.depositDataRoot
        );
    }
}
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	33	3577	538	1463	1576	136

Comments to Code 1463/1576 = 93%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	20	5172	547	98	4527	52

Tests to Code 4527/1576 = 287%