0.7

PancakeSwap Process Quality Review

Score: 82%

Overview

This is a PancakeSwap Process Quality Review completed on 8/1/2021. This is the second version of the report. The previous version is here. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Lucas of DeFiSafety. Check out our Telegram. The previous version of the review (0.7) is here.

The final score of the review is 82%, a solid 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

Chain

This section indicates the blockchain used by this protocol.

✓ **Chain: Binance Smart Chain**

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: 100%

They are available at website https://docs.pancakeswap.finance/code/smart-contracts, 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

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 56889 transactions a day on contract masterchef.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/pancakeswap

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 104 commits and 12 branches, this is a healthy 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

How to improve this score:

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

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

Answer: No

The PancakeSwap team is anonymous.

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.pancakeswap.finance/

How to improve this score:

Ensure that the white paper is available for download from your website or at least the software repository. Ideally update the whitepaper to meet the capabilities of your present application.

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

✓ Answer: Yes

PancakeSwap has impressive function documentation in their https://docs.pancakeswap.finance/code/smart-contracts documentation.

How to improve this score:

Write the document based on the deployed code. For guidance, refer to the SecurEth System Description Document.

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

✓ Answer: 100%

PancakeSwap has robust, easy to read, and well-written software function documentation that covers all of their code. Example: https://docs.pancakeswap.finance/code/smart-contracts/pancakeswap-exchange/factory-v2

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: 40%

Code examples are in the Appendix. As per the SLOC, there is 26% commenting to code (CtC). 15% was added for excellent commenting in Lottery.sol , and SousChef.sol.

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 explicit traceability for the vast majority of the documented functions.

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: 67%

As per the SLOC, there is 67% 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

How to improve this score:

This score can improved by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository.

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

✓ Answer: 90%

PancakeSwap has a code coverage of 90% as indicated in their Codecov.io report.

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

There are instructions to run the tests here.

How to improve this score:

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

ⓘ Answer: 70%

There is a GitHub Code Coverage report visible.

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%

There is no evidence of formal verification having been done.

16) Stress Testing environment (%)

⚠ Answer: 0%

There are no evident Kovan or Ropsten test net addresses.

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%

PakcakeSwap's core contracts have been audited by Certik, and twice by Slowmist.

PancakeSwap's lottery V2 has been audited twice, and most of the changes recommended have been implemented.

PancakeSwap's Peckshield Lottery V2 Audit

PancakeSwap's Slowmist Lottery V2 Audit

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: 100%

PancakeSwap has a 1,000,000$ bug bounty program with ImmuneFi, and it is an active program.

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: 40%

PancakeSwap's admin controls are not well labelled and spread across multiple directories.

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: 60%

b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30%

c) The capabilities for change in the contracts are described -- 30%

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%

Description relates to investments safety and updates in clear, complete non-software 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: 80%

Pause control is explained clearly but there is no evidence of regular testing.

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

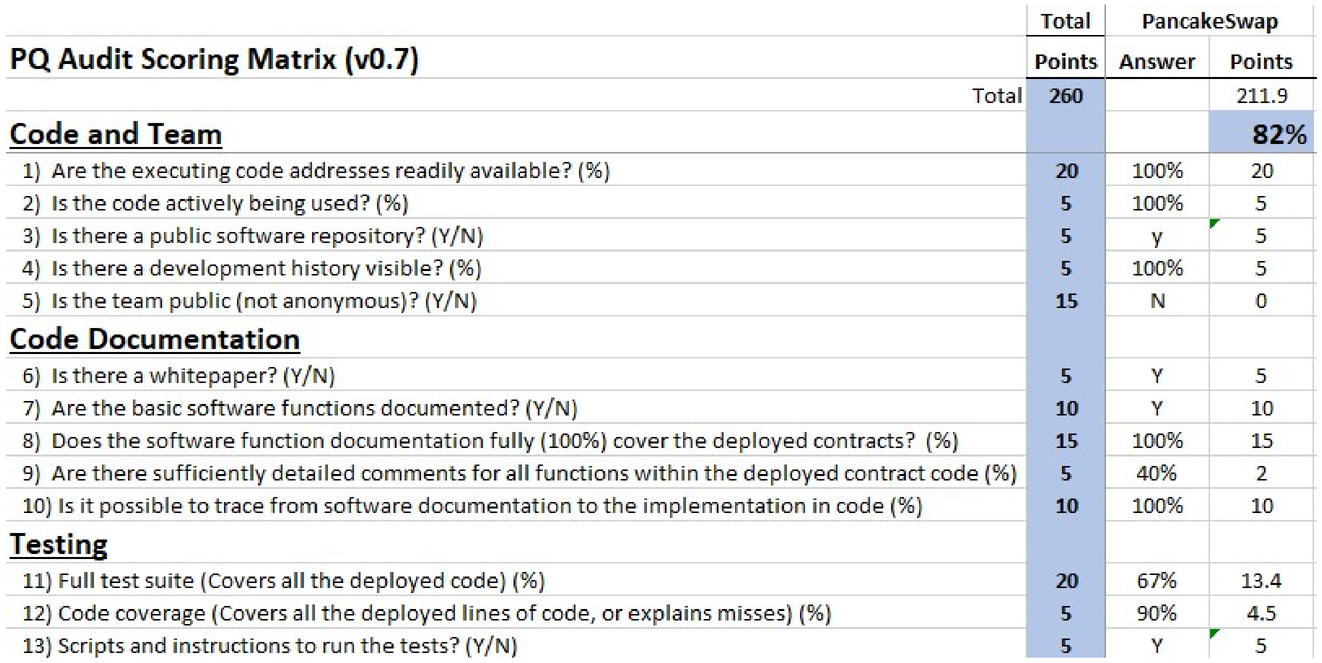
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

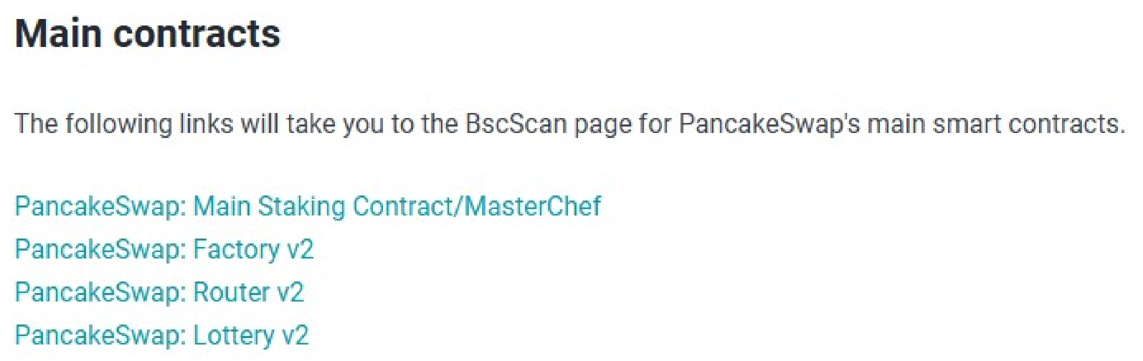
Scoring Appendix



	Total	PancakeSwap	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		211.9
Code and Team			**82%**
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**	N	0
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**	100%	15
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	40%	2
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**	67%	13.4
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	90%	4.5
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**	0%	0
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	100%	10
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	40%	2
20) Is the information clear and complete	**10**	60%	6
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of tests	**10**	80%	8
Section Scoring			
Code and Team	50	70%	
Documentation	45	93%	
Testing	50	60%	
Security	80	100%	
Access Controls	35	71%	

Executing Code Appendix

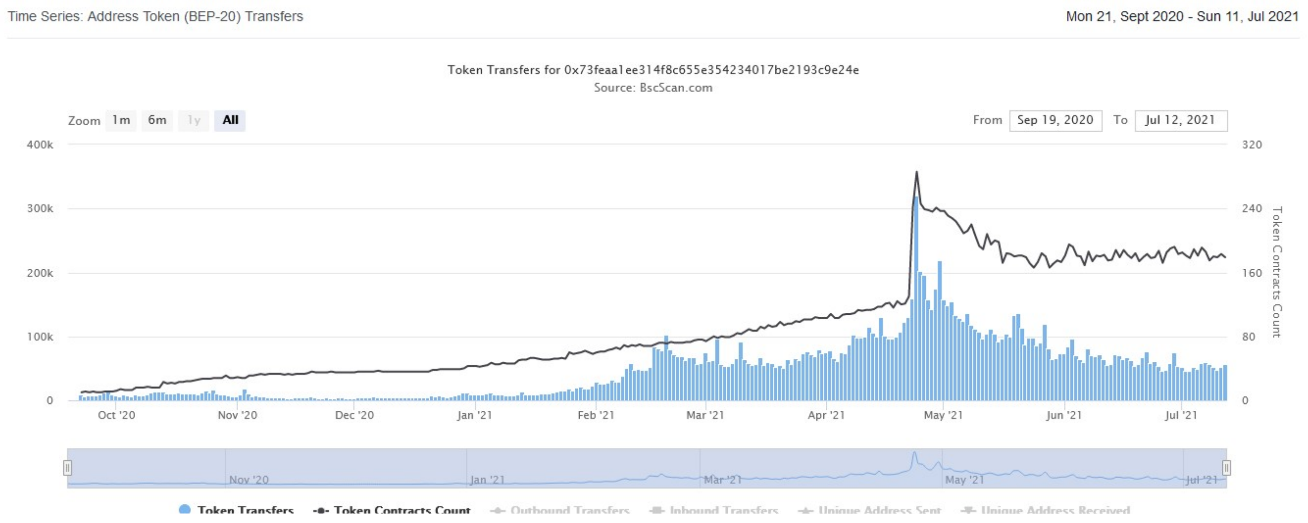


Main contracts

The following links will take you to the BscScan page for PancakeSwap's main smart contracts.

PancakeSwap: Main Staking Contract/MasterChef
PancakeSwap: Factory v2
PancakeSwap: Router v2
PancakeSwap: Lottery v2

Code Used Appendix



Example Code Appendix

```
pragma solidity 0.6.12;

import '@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol';
import '@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol';
import '@pancakeswap/pancake-swap-lib/contracts/token/BEP20/SafeBEP20.sol';
import '@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol';

// import "@nomiclabs/buidler/console.sol";

interface IWBNB {
    function deposit() external payable;
    function transfer(address to, uint256 value) external returns (bool);
    function withdraw(uint256) external;
}

contract BnbStaking is Ownable {
    using SafeMath for uint256;
    using SafeBEP20 for IBEP20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;     // How many LP tokens the user has provided.
        uint256 rewardDebt; // Reward debt. See explanation below.
        bool inBlackList;
    }

    // Info of each pool.
    struct PoolInfo {
        IBEP20 lpToken;           // Address of LP token contract.
        uint256 allocPoint;       // How many allocation points assigned to this pool. CAK
        uint256 lastRewardBlock;  // Last block number that CAKEs distribution occurs.
        uint256 accCakePerShare; // Accumulated CAKEs per share, times 1e12. See below.
    }

    // The REWARD TOKEN
    IBEP20 public rewardToken;

    // adminAddress
    address public adminAddress;



    // WBNB
    address public immutable WBNB;

    // CAKE tokens created per block.
    uint256 public rewardPerBlock;

    // Info of each pool.
    PoolInfo[] public poolInfo;

    // Info of each user that stakes LP tokens.
    mapping (address => UserInfo) public userInfo;
    // limit 10 BNB here
    uint256 public limitAmount = 10000000000000000000;
    // Total allocation poitns. Must be the sum of all allocation points in all pools.
    uint256 public totalAllocPoint = 0;
    // The block number when CAKE mining starts.
    uint256 public startBlock;
    // The block number when CAKE mining ends.
    uint256 public bonusEndBlock;

    event Deposit(address indexed user, uint256 amount);
    event Withdraw(address indexed user, uint256 amount);
    event EmergencyWithdraw(address indexed user, uint256 amount);

    constructor(
        IBEP20 _lp,
        IBEP20 _rewardToken,
        uint256 _rewardPerBlock,
        uint256 _startBlock,
        uint256 _bonusEndBlock,
        address _adminAddress,
        address _wbnb
    ) public {
        rewardToken = _rewardToken;
        rewardPerBlock = _rewardPerBlock;
        startBlock = _startBlock;
        bonusEndBlock = _bonusEndBlock;
        adminAddress = _adminAddress;
        WBNB = _wbnb;

        // staking pool
        poolInfo.push(PoolInfo({
            lpToken: _lp,
            allocPoint: 1000,
            lastRewardBlock: startBlock,
            accCakePerShare: 0
        }));

        totalAllocPoint = 1000;
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	32	3505	451	643	2411	270

Comments to Code 643/2411 = 26%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	7	773	83	15	675	0

Tests to Code 675/1000 = 67.5%