**WOW Coin Mining Dapp**

**Project Overview**

Wow Mining is a mining contract project based on ERC20 tokens. The architecture of the project is implemented using an upgradeable proxy pattern, which is divided into Delegate for implementation logic contract, Delegator for proxy contract and Storage for state variable storage contract.

This contract allows the project side to deposit ERC20 tokens into the contract and set the start block height for mining. When the block reaches the corresponding block height, automatic mining starts, and the contract automatically calculates the total amount of mining for this round (halving). At this time, the contract operator (can be set) is allowed to periodically withdraw a corresponding amount of tokens from the contract to a specified address (can be set) and generate logs. After one round of mining is completed, the next round will automatically start, and the total amount of tokens that can be mined in each round will be halved to mine until all the tokens in the contract are mined out.

**1.Functional Requirements**

1.Roles

Wow Mining has two roles:

● Owner: Contract deployment and upgrade.

● Operator: The address of this role has the permission to call the contract's withdraw() method and transfer tokens from the contract to the receiver address.

2.Features

Wow Mining has the following features:

● Initialize the contract. (Owner)

● Withdraw (mining) transfers tokens from the contract to the recipient address (operator).

● Query the remaining number of mineable tokens in the contract (anyone).

● Query the total number of mined tokens (anyone).

● Query the tuple information of all round information (anyone).

● Query the mining information of a certain round (anyone).

3.Use Cases

1. When creating the contract, Wow Mining needs to pass in the following information: total amount of mining, amount of mining per time, start block, ERC20 token address, and mining receiver address.

2. The administrator adds an operator through the addOperator() method.

3. After one round of mining is completed, the next round of mining starts automatically, and so on.

4. According to the mining progress, the operator periodically withdraws a fixed amount of tokens from the contract to the recipient address.

**2.Technical Requirements**

This project has been developed with Solidity language, using Truffle as a development environment. Javascript is the selected language for testing and scripting.

In addition, OpenZeppelin's libraries are used in the project. All information about the contracts library and how to install it can be found on their Github.

├── contracts

│ ├── test

│ │ └── MockToken.sol

│ └── wow

│ ├── WowMiningPoolDelegate.sol

│ ├── WowMiningPoolDelegator.sol

│ └── WowMiningPoolStorage.sol

├── coverage

├── migrations

├── package.json

├── test

│ └── WowMining.test.js

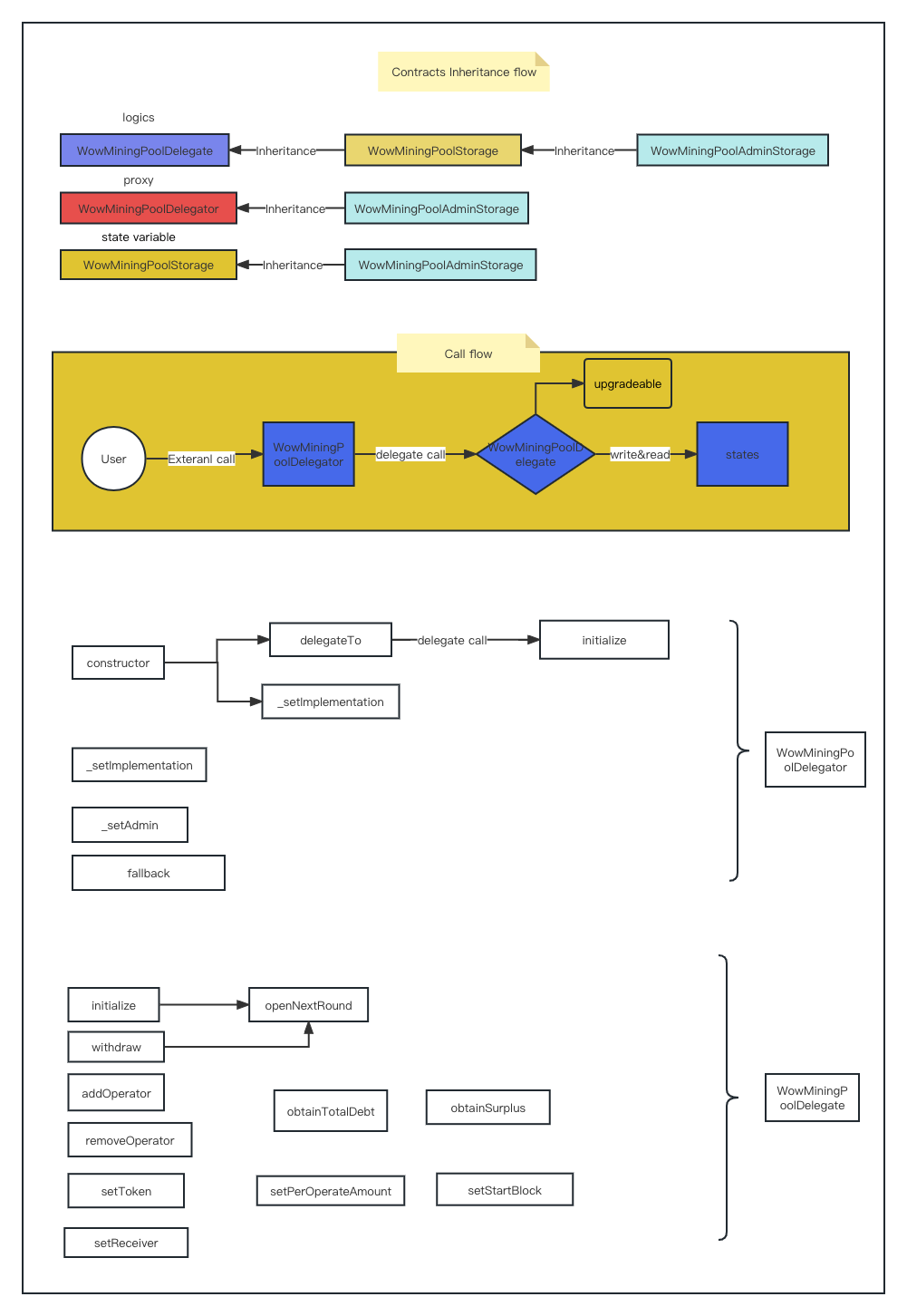
├── truffle-config.js

└── yarn.lock

The ./contracts/wow folder contains the main smart contracts for Wow Mining, where WowMiningPoolStorage.sol is the contract for storing the state variables, which includes two smart contracts WowMiningPoolStorage and WowMiningPoolAdminStorage. WowMiningPoolDelegator.sol is the proxy contract, mainly used to control the upgrade of contract logic, while WowMiningPoolDelegate.sol is the delegate contract, mainly used to write contract business logic.

In the ./test folder, WowMining.test.js provides unit tests for different contract methods. Run the unit tests with the command `truffle test ./test/WowMining.test.js`.

The project configuration information can be found in ./truffle-config.js, which contains the names and version information of some dependent modules, etc. For specific configuration, please refer to the official Truffle documentation.



2.1  **Architecture overview**

The following chart provides a general view of the proxy contract and logic contract structure and interactions between different functions.

For specific contract deployment or method usage, please refer to the unit tests of the contract.

A copy of the contract has been deployed on the BSC test network: https://testnet.bscscan.com/address/0x2D8b16c69EF9dDADF4db11843705ca35c856429A#readProxyContract