Project Overview

Arkania IDO Launchpad is a Solidity project based on ANIA tokens. It allows for admins to create projects which are listed for all users who have staked certain amount of ANIA tokens required for one of the three tier levels. Based on the tier level user can sign in to listed IDO and get whitelisted. Whitelisted users can buy tokens of the IDO for presale price and they receive the part of the tokens at the beginning and then the rest of the tokens are sent continuously.

1. Functional Requirements

1.1. Roles

Ania Stake contract has two roles:

- Owner: can change APY amount, can airdrop reward tokens for users with active stake
- User: can get total amount of staked tokens, can stake own tokens and lock into contract, can withdraw staked tokens and rewards, can get number of rewards, can get staked tokens with rewards

Ania Lottery contract has three roles:

- Owner: can set a new admin into the group of admins, can add an active stable coin into the group of stable coins for payments and withdraw project tokens locked in the contract
- Admin: Group of admins can perform these actions: remove another admin from the group, create project, update project, remove project, perform bulk add function into project whitelist, remove users from project whitelist, can draw lottery and define winners, change tier one requirements, change tier two requirements, change tier three requirements, change tier one price, change tier two price, change tier three price
- User: Can perform these actions: signup into the projects, sign out from the projects, can get his tier value according to current staked amount, can get the current stake cap of the project, can get the current raised amount of the project, can get the project and it’s data, can get the number of the users signed in the project, can get lottery winner, can get number of winners for the project, can check if user is signed in the project, can check if project is open for whitelisting into it, can check if the claim is available for the user before claim and pay for the tokens, can buy and claim tokens as a reward from lottery systém, can check balance of the tokens at address

1.2. Features

Arkania IDO Launchpad has the following features:

- Withdraw project tokens (Owner).
- Set admins (Owner)
- Set stable coins for payments (Owner)
- Create, update or delete project and lock its funds (tokens). (Admin)
- User bulk add into the project. (Admin)
- User bulk delete from the project (Admin)
- Draw lottery and set winners (Admin)

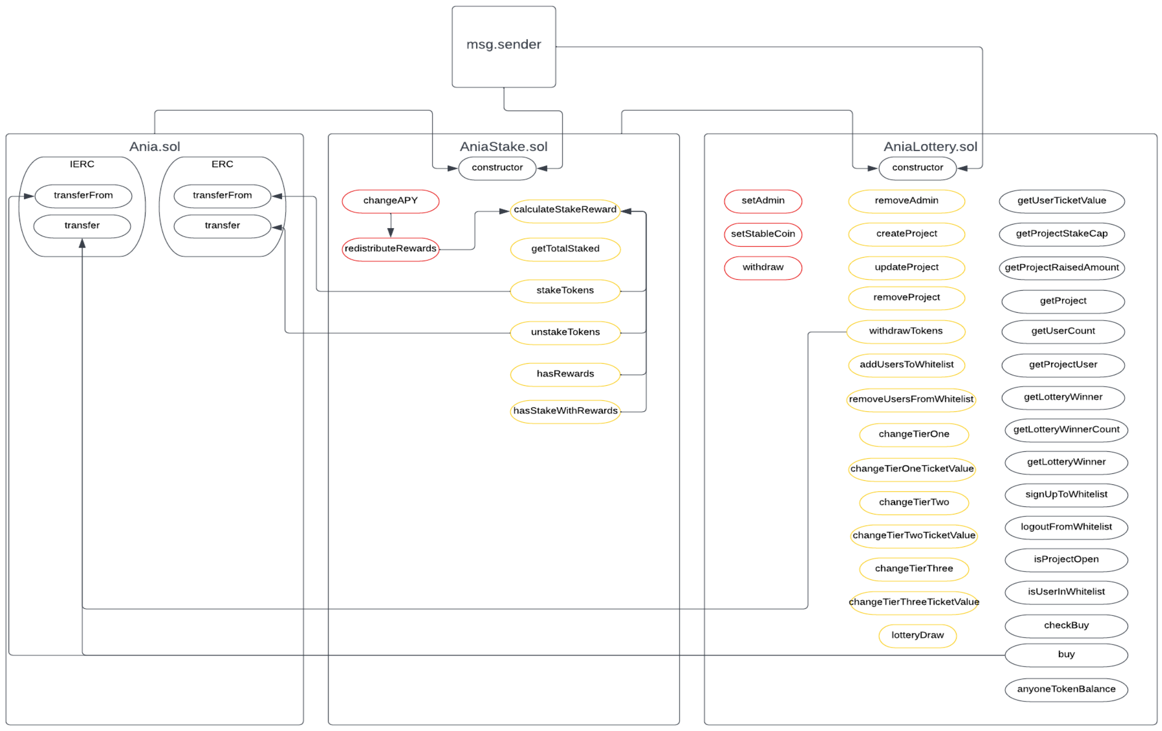
- Change tier system, their required number of staked tokens and price (Admin)
- Change APY and add rewards to currently staked ANIA tokens of certain user (Admin)
- Stake ANIA tokens (User)
- Unstake ANIA tokens (User)
- Sign in to the project (User)
- Sign out from the project (User)
- Buy and claim the project tokens if whitelisted (User)

1.3 Use Cases

1. Admin creates a new project and define its name, whitelisting period, presale price of the token, token contract address, billing address for payments, number of percentages of the tokens sent in first phase and raise goal of the project.
2. Owner of the project locks the number of tokens based on presale price and raise goal of the project.
3. Users stake their ANIA tokens to get into the one of the three tier levels.
4. Users sign up in projects they are interested in.
5. After whitelisting period ends, admin draws a lottery and set the certain number of the winners according to the raise goal of the project
6. Users who won the lottery can buy a certain amount defined by project owner in the first phase, for example 25% of the tokens.

2. Technical Requirements

2.1. Architecture Overview



2.2. Contract Information

Project uses two different contracts. First contract is only a stake system for ANIA tokens. This contract stores locked amount of Ania tokens which users locked into the contract and also owner's locked tokens for rewards. The second contract is used for creating IDOs and there is a lottery system for users who are signed into the certain IDO. With this contract we can check how many ANIA tokens certain user staked, which Tier value has and if he wins the lottery, user can also claim rewarded tokens from IDO for payment according to user's tier price, by default 250, 500 and 1000 tokens in stable coin added in the list of stable coins, for example BUSD.

2.2.1.AniaStake.sol

The main purpose of this contract is to lock and stake a user defined amount more than 0 and less than he owns. He can withdraw tokens anytime. Only the ANIA token is available to stake. ANIA tokens used for rewards have to be send on the contract manually.

2.2.1.1. Assets

AniaStake.sol contains two Structs:

- Stake: This object contains information about a stake.
    - address user – user address
    - uint256 amount - number of token in stake
    - Uint256 since – Timestamp for start staking
- Staker: Staker is a user who has active stakes
    - address user - user address

    - Stake address_stake -stake struct is used to represent the way we store stakes,

Besides the mentioned structs, the following entities are present in the project:

- stakers: A public array of type Struct Staker holds the users with stake.
- stakes: An address <-> uint256 mapping. Aims to store of the INDEX for the stakers in the stakes array assigned for a specific address.
- owner: ERC20 user address.
- totalStaked:
- apy: Initialize default value for reward

2.2.1.2. Functions
AniaStake has the following functions:

- constructor(ERC20 Ania): Handles configurations and sets related addresses.
- stakeTokens(uint256 _amount): Allows a user to stake Ania Token.
- unstakeTokens(uint256 _amount): Allows a user to unstake Ania Token.
- hasRewards() Show rewards from staking for user.
- hasStakeWithRewards(address _address): Show total balance for user. Stake coins with rewards.
- redistributeRewards(address[] calldata users): Allows admin to redistribute reward before change APY
- changeAPY(uint256 _value, address[] calldata users): Allows admin to change APY

2.2.2. AniaLottery.sol
The main purpose of this contract is to store projects and their locked tokens for lottery system. After the project is stored, users can sign in into the projects they are interested in only if they stake a required amount of ANIA tokens. According to their amount of staked tokens they are in corresponding Tier. After the lottery is drawn, winners can buy a part of the project tokens in presale price which are paid in defined stable coin, for example BUSD. Lottery can be drawn before the stake cap is fulfilled according to raise goal of the project. When the project raise goal is not fulfilled the leftover tokens of the contract can be returned only by the owner of the contract to avoid unauthorized takeover.

2.2.2.1. Assets
AniaLottery contains three Structs:

- Project: This object contains information about a project
    - uint id – id for project
    - string name – name for project
    - uint raiseGoal – raise goal for project
    - uint endDate – timestamp for close project
    - address contractAddress – contract address for project token
    - address billingAddress – billing address for project

    - uint firstPayoutInPercent – first pazout token in percent
    - uint256 tokenPrice – initial token price
    - bool draw – boolean if project is draw
- Whitelist: This structure holds information about a user who signed into project
    - uint projectId – id of project
    - uint signupDate – timestamp when user logged
    - address userAddress – user address who looged
- LotteryWinner: This object contains information about a users who won lottery.
    - uint projectId – id for project
    - address userAddress – user address
    - uint reward - Ticket Value for buy
    - bool claimed – boolen if user buy token

Besides the mentioned structs, the following entities are present in the project:

- `ProjectUserCount: An uint => uint` mapping to store count of project user
- `ProjectsWhitelist: An uint => Whitelist[]` object mapping for Whitelist
- `ProjectUserIndex: An uint => mapping(address => uint256)` mapping for users, who are signed in the ProjectsWhitelist at exact position
- `ProjectStakeCap: An uint => uint` mapping to store count for project stake
- `LotteryWinnerCount: An uint => uint` mapping to store count of lottery winner
- `LotteryWinners: An uint => LotteryWinner[]` object mapping to store winners
- `ProjectWinnerIndex: An uint => mapping(address => uint256)` mapping for users, who are winners in the LotteryWinners at exact position
- `ProjectRaisedAmount: An uint => uint` mapping to store raised amont
- `admins:An address => bool` mapping to store of user address who are admin
- `stableCoins: An address =>` object mapping to store of address for pay
- `Owner –` ERC20 user address.
- `TierOne –` Minimal stake value for first tier
- `TierOneTicketValue –` Ticket Value for buy for first tier
- `TierTwo –` Minimal stake value for second tier
- `TierTwoTicketValue –` Ticket Value for buy for second tier

- `TierThree` – Minimal stake value for three tier
- `TierThreeTicketValue` – Ticket Value for buy for three tier
- `decimals` - decimals

2.2.2.2. Modifiers

Ania Lottery contract has the following modifiers:

- onlyOwner:Check if user is creator
- onlyAdmin: Checks if user is owner or admin

2.2.2.3. Functions

AniaLottery has the following functions:

- constructor(ERC20 AniaStake): Handles configurations and sets related addresses.
- setAdmin(address _admin, bool isAdmin): Allows owner to set user as admin
- removeAdmin (address adminAddress): Allows owner to remove user from admins
- setStableCoin(address _address, bool isActive): Allows owner to add tokens for pay
- createProject(uint projectId, string calldata projectName, uint raiseGoal, uint endDate, address contractAddress, address billingAddress, uint firstPayoutInPercent, uint256 tokenPrice): Allows admin create project
- updateProject(uint projectId, string calldata projectName, uint raiseGoal, uint endDate, address contractAddress, address billingAddress, uint firstPayoutInPercent, uint256 tokenPrice): Allows admin update project
- removeProject(uint projectId): Allows admin remove project
- withdraw(): Allows owner withdraw tokens
- withdrawTokens(uint projectId, address recipient): Allows admin send tokens from specific project to specific address
- addUsersToWhitelist(uint projectId, address[] calldata users, bool checkEndDate): Allows admin add users to Project list
- removeUsersFromWhitelist(uint projectId, address[] calldata users): Allows admin remove users from Project list
- getUserTicketValue(address _address): Allow user get actual Ticket Value for his stake
- getProjectStakeCap(uint projectId): Allow user get actual stake value for specific project
- getProjectRaisedAmount(uint projectId): Allow user get actual raised value for specific project
- lotteryDraw(uint projectId, address[] calldata users) : Allow admin draw lottery
- getProject(uint projectId): Allow user show Project Info
- getUserCount(uint projectId): Allow user show count of project users
- getProjectUser(uint projectId, address userAddress): Allow user show project users

- getLotteryWinner(uint projectId, address userAddress): Allow user show project winner
- getLotteryWinnerCount(uint projectId): Allow user show count of project winner
- changeTierOne(uint _value): Allow admins change minimal stake value for first tier
- changeTierTwo(uint _value): Allow admins change minimal stake value for second tier
- changeTierThree(uint _value): Allow admins change minimal stake value for third tier
- changeTierOneTicketValue(uint _value): Allow admins change Ticket Value for buy for first tier
- changeTierTwoTicketValue(uint _value): Allow admins change Ticket Value for buy for second tier
- changeTierThreeTicketValue(uint _value): Allow admins change Ticket Value for buy for third tier
- signUpToWhitelist(uint projectId): Allows users to sign up to Project list for lottery.
- logoutFromWhitelist(uint projectId): Allows users to logout from Project list for lottery.
- isProjectOpen(uint projectId): Check if project is open.
- isUserInWhitelist(uint projectId): Check if user is on project list.
- checkBuy(uint projectId, uint256 tokensToBuy): Check if user is winner.
- buy(uint projectId, uint pay, address tokenForPayContractAddress): Allow users to buy project token if is winner.
- anyoneTokenBalance(address tokenContractAddress, address userAddress): Check balance specific token for specific address