CryptoIndex Smart Contract Documentation

Overview

The `MemeIndex` smart contract allows influencers to create indices of meme tokens with specified percentages. Users can then buy these indices with BNB, and the contract will handle the conversion and storage of the tokens. Users can later sell their indices, and the contract will handle the conversion of tokens back to BNB, deducting a fee.

Key Components

1. IPancakeRouter Interface

- Interface for interacting with PancakeSwap's router contract, enabling token swaps.

2. Structs

- `Index`: Stores the tokens and their respective percentages in an index.
- `UserIndex`: Stores user-specific data for each index they have bought, including the amount of BNB spent and the amounts of each token.

3. Mappings

- `indices`: Maps index IDs to their respective `Index` struct.
- `userIndexes`: Maps user addresses to another mapping of index IDs to `UserIndex` structs.

4. Events

- `IndexCreated`: Emitted when a new index is created.
- `IndexBought`: Emitted when a user buys an index.
- `IndexSold`: Emitted when a user sells an index.
- `FeeCollected`: Emitted when a fee is collected from a sale.

Functions

Constructor

solidity constructor(address _pancakeRouter)
- Initializes the contract with the PancakeRouter)
- Initializes the contract with the PancakeRouter address.
Parameters:

- `_pancakeRouter`: The address of the PancakeRouter contract.

createIndex

solidity function createIndex(address[] calldata tokens, uint256[] calldata percentages) external onlyOwner
- Allows the owner to create a new index with specified tokens and percentages.
- Validates that the total percentage equals 100% and that there are exactly 5 tokens.
- Stores the index in the `indices` mapping.
- Emits `IndexCreated` event.

Parameters:
- `tokens`: The addresses of the tokens in the index.
- `percentages`: The percentages of the total investment allocated to each token.

buyIndex

solidity function buyIndex(uint256 indexId) external payable nonReentrant
- Allows a user to buy an index by sending BNB.
- Converts BNB to the specified tokens using PancakeSwap and stores them in the contract.
- Stores the user's investment details in the `userIndexes` mapping.
- Emits `IndexBought` event.

sellIndex

solidity function sellIndex(uint256 indexId, uint256 bnbAmount) external nonReentrant
- Allows a user to sell a specified amount of an index they own.
- Calculates the proportion of the index to sell based on the specified BNB amount.
- Converts the tokens back to BNB using PancakeSwap.
- Deducts a fee and transfers the remaining BNB to the user.
- Updates or deletes the user's investment details in the `userIndexes` mapping.
- Emits `IndexSold` and `FeeCollected` events.

Parameters:
- `indexId`: The ID of the index the user wants to sell.
- `bnbAmount`: The amount of BNB worth of the index to sell.

updateFeePercentage

solidity function updateFeePercentage(uint256 newFeePercentage) external onlyOwner
- Allows the owner to update the fee percentage for selling indices.
- Validates that the new fee percentage is less than or equal to 100%.

Parameters:
- `newFeePercentage`: The new fee percentage to set.

getUserIndexes

solidity function getUserIndexes(address user) external view returns (UserIndex[] memory)
- Returns an array of `UserIndex` structs representing the indices a user has bought.

Parameters:
- `user`: The address of the user whose indices are to be retrieved.

Usage

1. Creating an Index

- The contract owner calls `createIndex` with the addresses of 5 tokens and their respective percentages summing to 100.
- This creates a new index and emits an `IndexCreated` event.

2. Buying an Index

- A user calls `buyIndex` with the ID of the desired index and sends BNB.
- The contract swaps BNB for the specified tokens according to the index's percentages.
- The user's investment is stored in the `userIndexes` mapping, and an `IndexBought` event is emitted.

3. Selling an Index

- A user calls `sellIndex` with the ID of the index and the amount of BNB they want to sell.
- The contract calculates the proportion of the index to sell, swaps the tokens back to BNB, and transfers the BNB (minus a fee) to the user.
- The user's investment details are updated or deleted as necessary, and `IndexSold` and `FeeCollected` events are emitted.

4. Updating the Fee Percentage

- The contract owner calls `updateFeePercentage` with the new fee percentage.
- The fee percentage is updated if it is less than or equal to 100

5. **Retrieving User Indices**

- Anyone can call `getUserIndexes` with a user's addres to retrieve an array of `UserIndex` structs representing the indices the user has bought.

Example 1.

1. Creating an Index

solidity contract.createIndex(["0xToken1", "0xToken2", "0xToken3", "0xToken4", "0xToken5"], [20, 20, 20, 20, 20]);

2. Buying an Index

solidity contract.buyIndex{value: 1 ether}(0);

3. Selling an Index

solidity contract.sellIndex(0, 0.5 ether);

4. Updating the Fee Percentage

solidity contract.updateFeePercentage(5);

5. Retrieving User Indices

solidity UserIndex[] memory indices = contract.getUserIndexes(userAddress);