**Purpose**

This Solidity contract implements a simple staking system. Users can stake their tokens for a certain period of time and earn rewards based on the amount of tokens staked and the staking period.

**Usage**

To use the contract, users first need to stake their tokens. This can be done by calling the stake() function. The stake() function takes a single parameter, \_amount, which is the amount of tokens to be staked.

Once a user has staked their tokens, they can start earning rewards. Rewards are distributed on a regular basis by the contract owner. To distribute rewards, the contract owner needs to call the distributeRewards() function.

Users can withdraw their staked tokens and rewards at any time by calling the withdraw() function.

**Function Documentation**

**Constructor**

constructor(uint \_stakingPeriod, uint \_totalRewards) {

owner = msg.sender;

stakingPeriod = \_stakingPeriod;

totalRewards = \_totalRewards;

}

The constructor is used to initialize the contract. It takes two parameters: \_stakingPeriod, which is the staking period in seconds, and \_totalRewards, which is the total amount of rewards to be distributed.

**Stake**

function stake(uint \_amount) public {

require(\_amount > 0, "Amount should be greater than 0");

require(userBalances[msg.sender] >= \_amount, "Insufficient balance");

userBalances[msg.sender] -= \_amount;

stakedBalances[msg.sender] += \_amount;

stakingStartTimes[msg.sender] = block.timestamp;

// Add the user to the list if they're not already there

if (stakedBalances[msg.sender] > 0) {

users.push(msg.sender);

}

}

The stake() function allows users to stake their tokens. It takes a single parameter, \_amount, which is the amount of tokens to be staked.

**CalculateRewards**

function calculateRewards(address \_user) public view returns (uint) {

uint timeStaked = block.timestamp - stakingStartTimes[\_user];

uint stakedAmount = stakedBalances[\_user];

return (stakedAmount \* timeStaked) / stakingPeriod;

}

The calculateRewards() function calculates the amount of rewards that a user has earned. It takes a single parameter, \_user, which is the address of the user to calculate rewards for.

**DistributeRewards**

function distributeRewards() public {

require(msg.sender == owner, "Only owner can distribute rewards");

for (uint i = 0; i < users.length; i++) {

address user = users[i];

uint rewards = calculateRewards(user);

userBalances[user] += rewards;

}

}

The distributeRewards() function distributes rewards to all users who have staked their tokens. It can only be called by the contract owner.

**Withdraw**

function withdraw() public {

require(stakedBalances[msg.sender] > 0, "Nothing staked");

uint rewards = calculateRewards(msg.sender);

userBalances[msg.sender] += rewards;

userBalances[msg.sender] += stakedBalances[msg.sender];

stakedBalances[msg.sender] = 0;

}

The withdraw() function allows users to withdraw their staked tokens and rewards. It can only be called by users who have staked tokens.