// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract DecentralizedLottery {

address public manager;

address[] public players;

uint public entryFee;

uint public maxPlayers;

uint public lotteryEndTime;

uint public winnerIndex;

address public winner;

event PlayerEntered(address indexed player);

event WinnerSelected(address indexed winner, uint amount);

constructor(uint \_entryFee, uint \_maxPlayers, uint \_durationMinutes) {

manager = msg.sender;

entryFee = \_entryFee;

maxPlayers = \_maxPlayers;

lotteryEndTime = block.timestamp + (\_durationMinutes \* 1 minutes);

}

modifier onlyManager() {

require(msg.sender == manager, "Only manager can call this");

\_;

}

modifier lotteryActive() {

require(block.timestamp < lotteryEndTime, "Lottery has ended");

\_;

}

function enterLottery() public payable lotteryActive {

//require(msg.value == entryFee, "Incorrect entry fee");

require(players.length <= maxPlayers, "Player limit reached");

players.push(msg.sender);

emit PlayerEntered(msg.sender);

// Automatically pick winner if max players reached

if (players.length == maxPlayers) {

pickWinner();

}

}

function pickWinner() public onlyManager {

require(players.length > 0, "No players in the lottery");

require(block.timestamp >= lotteryEndTime || players.length == maxPlayers, "Lottery still active");

// Simple pseudo-random selection (not secure for high-value lotteries)

winnerIndex = uint(keccak256(abi.encodePacked(block.timestamp, block.difficulty, players))) % players.length;

winner = players[winnerIndex];

uint prizeAmount = address(this).balance;

(bool success, ) = winner.call{value: prizeAmount}("");

require(success, "Transfer failed");

emit WinnerSelected(winner, prizeAmount);

// Reset the lottery

while (players.length > 0) {

players.pop();

}

lotteryEndTime = block.timestamp + 1 days;

}

function getPlayers() public view returns (address[] memory) {

return players;

}

function withdraw() public onlyManager {

// Allow manager to withdraw funds if needed (fails if there are players)

require(players.length == 0, "Players exist, cannot withdraw");

(bool success, ) = manager.call{value: address(this).balance}("");

require(success, "Withdraw failed");

}

receive() external payable {

enterLottery();

}

}