INTEGRAL

Interview Question: E

Overview

Below you will find a smart contract written in Solidity. There are a number of issues with the contract below, including:

1. Coding bugs
2. Conventions and best practices
3. Security
4. Memory and gas usage
5. Architecture and implementation

Objectives

1. **Review and correct the smart contract below:** Please discuss your findings and thought process with the interviewer. If you find an issue but do not have an immediate solution, you should still identify it to the interviewer and explain why the code is problematic.

```
// SPDX-License-Identifier: GPL-3.0-or-later

pragma solidity ^0.7.5;

import '@openzeppelin/contracts/token/ERC20/ERC20.sol';
import '@openzeppelin/contracts/math/SafeMath.sol';

contract Lottery {
    address public owner;

    mapping (address => uint256) bets;
    mapping (address => bool) collected;
    mapping (uint256 => address payable) users;

    uint256 counter;
    uint256 reward;
    uint256 rewards;
    address public token;

    constructor(uint256 _reward, address _token) {
        owner = msg.sender;
        reward = _reward;
        token = _token;
    }

    function setOwner(address _owner) public {
        owner = _owner;
    }

    function setReward(uint256 _reward) public {
        require(msg.sender == owner, "FORBIDDEN");
        reward = _reward;
    }

    function bet(uint256 _bet) public {
        require(msg.value > reward, "INVALID_VALUE");
        bets[msg.sender] = _bet;
        counter++;
        users[counter] = msg.sender;
    }

    function collectReward(address to) public {
        if (bets[msg.sender] == block.number % 10) {
            ERC20(token).transfer(to, reward);
            collected[to] = true;
            rewards = rewards + reward;
        }
    }

    function rewardUsers() payable public {
        require(msg.sender == owner, "FORBIDDEN");
        for (uint32 i = 0; i < counter; i++) {
            users[i].send(msg.value / counter);
        }
    }
}
```