// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract PollingSystem {

struct Poll {

string title;

string[] options;

uint endTime;

mapping(uint => uint) votes; // option index => vote count

mapping(address => bool) hasVoted; // tracks if a voter has voted

bool exists;

}

uint public pollIdCounter;

mapping(uint => Poll) public polls;

// Create a new poll

function createPoll(string memory \_title, string[] memory \_options, uint \_durationInMinutes) external {

require(\_options.length > 1, "At least 2 options required");

Poll storage poll = polls[pollIdCounter];

poll.title = \_title;

poll.options = \_options;

poll.endTime = block.timestamp + (\_durationInMinutes \* 1 minutes);

poll.exists = true;

pollIdCounter++;

}

// Vote on a poll

function vote(uint \_pollId, uint \_optionIndex) external {

require(polls[\_pollId].exists, "Poll does not exist");

require(block.timestamp < polls[\_pollId].endTime, "Poll has ended");

require(!polls[\_pollId].hasVoted[msg.sender], "You have already voted");

require(\_optionIndex < polls[\_pollId].options.length, "Invalid option");

polls[\_pollId].votes[\_optionIndex]++;

polls[\_pollId].hasVoted[msg.sender] = true;

}

// View poll options

function getOptions(uint \_pollId) external view returns (string[] memory) {

require(polls[\_pollId].exists, "Poll does not exist");

return polls[\_pollId].options;

}

// Get votes for each option (only after poll ends)

function getVotes(uint \_pollId) external view returns (uint[] memory) {

require(polls[\_pollId].exists, "Poll does not exist");

require(block.timestamp >= polls[\_pollId].endTime, "Poll is still active");

uint optionCount = polls[\_pollId].options.length;

uint[] memory result = new uint[](optionCount);

for (uint i = 0; i < optionCount; i++) {

result[i] = polls[\_pollId].votes[i];

}

return result;

}

// Get winning option (after poll ends)

function getWinningOption(uint \_pollId) external view returns (string memory winningOption) {

require(polls[\_pollId].exists, "Poll does not exist");

require(block.timestamp >= polls[\_pollId].endTime, "Poll is still active");

uint maxVotes = 0;

uint winnerIndex = 0;

for (uint i = 0; i < polls[\_pollId].options.length; i++) {

if (polls[\_pollId].votes[i] > maxVotes) {

maxVotes = polls[\_pollId].votes[i];

winnerIndex = i;

}

}

return polls[\_pollId].options[winnerIndex];

}

}