```
// SPDX-License-Identifier: MIT
pragma solidity ^0.7.0;

contract Voting {
    // Structure to represent a candidate
    struct Candidate {
        string name;
        uint voteCount;
    }

    // Structure to represent a voter
    struct Voter {
        bool isRegistered;
        bool hasVoted;
        uint votedCandidateId;
    }

    address public admin;
    bool public votingOpen;
    uint public totalVotes;
    Candidate[] public candidates;
    mapping(address => Voter) public voters;

    event NewCandidate(uint candidateId, string name);
    event Voted(address voter, uint candidateId);

    modifier onlyAdmin() {
        require(msg.sender == admin, "Only the admin can perform this action");
        _;
    }
```

```
modifier canVote() {
    require(votingOpen, "Voting is closed");
    require(voters[msg.sender].isRegistered, "You are not registered to vote");
    require(!voters[msg.sender].hasVoted, "You have already voted");
    _;
}

constructor() {
    admin = msg.sender;
    votingOpen = false;
}

function openVoting() public onlyAdmin {
    votingOpen = true;
}

function closeVoting() public onlyAdmin {
    votingOpen = false;
}

function registerVoter(address voterAddress) public onlyAdmin {
    require(!voters[voterAddress].isRegistered, "Voter is already registered");
    voters[voterAddress].isRegistered = true;
}

function addCandidate(string memory name) public onlyAdmin {
    uint candidateId = candidates.length;
    candidates.push(Candidate(name, 0));
    emit NewCandidate(candidateId, name);
}
```

```
    function vote(uint candidateId) public canVote {
        require(candidateId < candidates.length, "Invalid candidate index");
        voters[msg.sender].hasVoted = true;
        voters[msg.sender].votedCandidateId = candidateId;
        candidates[candidateId].voteCount++;
        totalVotes++;
        emit Voted(msg.sender, candidateId);
    }

    function getNumCandidates() public view returns (uint) {
        return candidates.length;
    }

    function getCandidate(uint index) public view returns (string memory, uint) {
        require(index < candidates.length, "Invalid candidate index");
        return (candidates[index].name, candidates[index].voteCount);
    }
}
```