Proposal storage

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.20;

contract Voting {
    // Define the Proposal struct
    struct Proposal {
        address target;
        bytes callData;
        uint yesCount;
        uint noCount;
    }

    // Public array to store all proposals
    Proposal[] public proposals;

    // External function to create a new proposal
    function newProposal(address _target, bytes calldata _callData) external {
        proposals.push(Proposal({
            target: _target,
            callData: _callData,
            yesCount: 0,
            noCount: 0
        }));
    }
}
```

Cast a Voting

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.20;

contract Voting {
    // Define the Proposal struct
    struct Proposal {
        address target;
        bytes callData;
        uint yesCount;
        uint noCount;
    }

    // Public array to store all proposals
    Proposal[] public proposals;

    // External function to create a new proposal
    function newProposal(address _target, bytes calldata _callData) external {
        proposals.push(Proposal({
            target: _target,
            callData: _callData,
            yesCount: 0,
            noCount: 0
        }));
    }

    // External function to cast a vote on a proposal
```

```
    function castVote(uint proposalId, bool support) external {
        Proposal storage proposal = proposals[proposalId];
        if (support) {
            proposal.yesCount++;
        } else {
            proposal.noCount++;
        }
    }
}
```

Multiple Votes

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.20;

contract Voting {
    struct Proposal {
        address target;
        bytes callData;
        uint yesCount;
        uint noCount;
    }

    Proposal[] public proposals;

    // Tracks whether an address has voted on a specific proposal
    mapping(uint => mapping(address => bool)) public hasVoted;
    // Tracks the vote choice of an address on a specific proposal
```

```
mapping(uint => mapping(address => bool)) public voteChoice;

function newProposal(address _target, bytes calldata _callData) external {
    proposals.push(Proposal({
        target: _target,
        callData: _callData,
        yesCount: 0,
        noCount: 0
    }));
}

function castVote(uint proposalId, bool support) external {
    Proposal storage proposal = proposals[proposalId];

    if (hasVoted[proposalId][msg.sender]) {
        // Voter has already voted
        bool previousVote = voteChoice[proposalId][msg.sender];
        if (previousVote != support) {
            // Adjust counts due to changed vote
            if (previousVote) {
                proposal.yesCount--;
                proposal.noCount++;
            } else {
                proposal.noCount--;
                proposal.yesCount++;
            }
            // Update stored vote
```

```
            voteChoice[proposalId][msg.sender] = support;
        }
        // If vote hasn't changed, do nothing
    } else {
        // First-time voter on this proposal
        if (support) {
            proposal.yesCount++;
        } else {
            proposal.noCount++;
        }
        hasVoted[proposalId][msg.sender] = true;
        voteChoice[proposalId][msg.sender] = support;
    }
  }
}
```