**Completed Modules**

* **Module 1: Proposal Creation**
  + **Users with ≥10 delegated votes can submit proposals.**
  + **Proposal must specify its type: Routine or Strategic.**
* **Module 2: Voting Weight & Strategy *(Bonus included)***
  + **Implements Quadratic Voting (vote weight = tokens committed).**
  + **Strategic proposals use Weighted Random Dictatorship:  
    proposals are selected probabilistically based on vote weight.**
* **Module 3: Eligibility Criteria**
  + **Only users with non-zero delegated token voting power can vote.**
  + **Checked via token.getVotes(msg.sender).**
* **Module 4: Voting Lifecycle**
  + **Proposal state (Pending, Active, Succeeded, Defeated, Executed) is dynamically determined from timestamps.**
  + **No manual intervention required.**
* **Module 5: Proposal Success Criteria**
  + **A proposal passes if:**
    - **yesVotes > noVotes, and**
    - **total quadratic votes ≥ threshold (e.g., 5).**
  + **Prevents low-turnout approvals.**

**Remaining Modules**

* **Module 6: Proposal Execution *(Bonus: multi-sig)***
  + **Current version: basic callData execution after proposal passes.**
  + **To do: add multi-signature requirement (e.g., 2-of-3 confirmations) before execution.**
* **Module 7: Proposal Struct Design**
  + **Core data structures implemented using mappings.**
  + **To do: optional refactoring for modularity, gas efficiency, or off-chain readability.**
* **Module 8: Assembly/Yul Integration *(Required)***
  + **No assembly {} block included yet.**
  + **To do: insert at least one line of valid Yul for compliance (e.g., in sqrt() or executeProposal()).**

**Detailed Description**

**Governance Mechanism Design (Modules 1–5)**

**1.Proposal creation** is implemented with an emphasis on structured input and categorized intentions. Each proposal must be submitted by an eligible token holder—defined as having at least 10 delegated governance tokens—thereby ensuring that only participants with a baseline level of stake in the system are able to initiate governance decisions.

In addition to the usual metadata (description, callData, and timing), each proposal must specify its type: either *Routine* or *Strategic*. This classification determines whether the proposal will follow a standard success-vote model or trigger the alternative selection mechanism detailed below.

**2.Voting weights and strategy** follow a **quadratic voting model**. In this model, the effective voting weight for any participant is the square root of the number of tokens they choose to commit. This design captures the idea that as a participant casts more votes (i.e., commits more tokens), the marginal influence of each additional token diminishes. This discourages the domination of outcomes by wealthy participants while still allowing them greater voice than others.

Moreover, in the case of strategic proposals, the quadratic votes received by each proposal are interpreted not as definitive decisions but as probabilistic weights. That is, once the voting period ends, a ***Weighted Random Dictatorship*** mechanism is used to select a single proposal for execution, where each candidate's chance of being selected is proportional to its received quadratic vote total. This mechanism introduces a probabilistic fairness that respects community preference while preventing the deterministic domination of the highest-vote option (Gibbard, 1977; Fishburn, 1982).

**3.Eligibility criteria** are enforced such that only users who hold delegated token voting power can participate in the voting process. This is operationalized via a check that the user’s voting power, as returned by getVotes(msg.sender), is strictly positive. This ensures that governance participation remains limited to active, legitimate DAO members.

**4.Voting lifecycle management** is implemented through time-dependent state tracking. Each proposal includes voteStart and voteEnd timestamps, and its state is computed dynamically based on the current block timestamp.

Proposals progress through a standard sequence: Pending (before voting begins), Active (during the voting window), Succeeded or Defeated (based on vote results after voting closes), and Executed (once an action is performed). This temporal structure allows external contracts or front-end clients to interact with proposals according to their state without requiring manual state transitions.

**5.Proposal success criteria** follow a two-part rule. First, the number of yes votes must exceed the number of no votes. Second, the total number of quadratic votes cast must exceed a defined minimum threshold (e.g., 5). This double condition ensures that proposals are not only net-positive in support but also sufficiently participated in. This addresses the common DAO problem of low-turnout approvals and encourages proposers to build consensus before submitting proposals.