### Updated Document: Reward and Burn Logic and Workflow

#### **Overview**

The Reward and Burn Logic governs the alternating daily cycles of token rewards and supply reduction through burns. Leveraging a hybrid on/off-chain architecture ensures efficiency, scalability, and compliance with the eligibility rules. This document reflects the latest workflow updates, incorporating feedback and revisions from the development team.

### ****Step 1: Hybrid Approach for Wallet Selection****

1. **Data Collection (Off-Chain):**
   * Retrieve all wallet addresses holding tokens daily.
   * Gather relevant metadata, such as:
     + Balances
     + Transaction history
     + Cooldown status
2. **Eligibility Filtering (Off-Chain):**
   * Apply filters to remove ineligible wallets:
     + **Minimum Token Balance Check:** Exclude wallets with balances below the threshold (e.g., 1,000 tokens).
     + **Cooldown Status Check:** Exclude wallets currently flagged for cooldown due to recent sales.
     + **Blacklist Exclusion:** Remove contract wallets (e.g., dev, marketing, team) and known exploit wallets.
3. **Random Selection (On-Chain):**
   * Use Chainlink VRF to randomly select **10% of eligible wallets**, capped at 1,000 wallets.
   * Pass the filtered wallet data to Chainlink for randomness verification and selection.
4. **Reward Distribution (On-Chain):**
   * Distribute **1% of the reward wallet** equally among selected wallets.

### ****Logical Flowchart: Reward/Burn Wallet Process****

**Start** ⬇

1. Pull Daily Wallet Data (Off-Chain)  
   ⬇
2. Apply Eligibility Filters (Off-Chain):
   * **Min Token Balance:**
     + Yes → Continue
     + No → Exclude
   * **Cooldown Active:**
     + No → Continue
     + Yes → Exclude
   * **Blacklisted:**
     + No → Continue
     + Yes → Exclude  
       ⬇
3. Submit Eligible Wallets to Chainlink VRF (On-Chain):
   * Generate Random Selection of Wallets  
     ⬇
4. Distribute Rewards or Execute Burn (On-Chain):
   * **Reward Day:** Distribute 1% of the reward wallet equally to selected wallets.
   * **Burn Day:** Burn 1% of the reward wallet to reduce supply.  
     ⬇  
     **End**

### ****Remaining Tasks****

1. **Eligibility Criteria Refinement:**
   * Finalize the minimum token balance (e.g., 1,000 tokens).
   * Define cooldown exclusions based on wallet activity (e.g., sale percentages triggering cooldown).
   * Solidify criteria for blacklisted wallets (e.g., team wallets, exploit wallets).
2. **Random Selection Logic:**
   * Confirm integration of Chainlink VRF.
   * Ensure the off-chain filtering process includes:
     + Wallet data retrieval
     + Application of eligibility filters
     + Creation of the eligible wallets pool for Chainlink VRF
3. **Reward Distribution Logic:**
   * Develop daily distribution logic:
     + Fetch random wallet selection from Chainlink VRF.
     + Calculate the reward share per wallet.
     + Execute transfers to distribute rewards.
4. **Burn Logic:**
   * Design the burn function:
     + Calculate 1% of the reward wallet for burn days.
     + Execute the burn transaction and log it on-chain.
5. **Automation and Timing:**
   * Develop logic for alternating between reward and burn days.
   * Automate process triggers based on a daily schedule.
6. **Off-Chain Workflow:**
   * Develop scripts for:
     + Daily wallet data retrieval
     + Eligibility filtering
     + Submission to Chainlink VRF
7. **Testing Framework:**
   * Create unit tests for:
     + Eligibility filters
     + Random selection via Chainlink VRF
     + Reward distribution
     + Burn logic
   * Mock integration with Chainlink VRF for testing.
8. **Transparency and Logs:**
   * Add event logs for:
     + Eligible wallets selected for rewards
     + Amount of tokens distributed or burned daily
     + Randomness request and result validation
   * Ensure all logs are accessible for audit purposes.

### ****Updated Prioritized Development Plan****

1. **Core Token Contract:**
   * Foundation of all functionality, enabling basic token operations.
2. **Reward/Burn Contract:**
   * Governs daily reward and burn cycles, ensuring alignment with tokenomics.
3. **Cooldown Contract:**
   * Enforces disciplined behavior through cooldown mechanisms.
4. **Off-Chain Analytics Script:**
   * Provides wallet filtering and prepares the eligible wallets list for Chainlink VRF.
5. **AI Middleware:**
   * Adds intelligence to adjust reward/burn parameters and engage users dynamically.
6. **Governance Contract:**
   * Enables community participation and decentralized decision-making.
7. **Community Engagement Tools:**
   * Enhances user interaction through notifications, leaderboards, and trivia.

### ****Logical Flowchart for Wallet Selection Process****

**Start** ⬇

1. **Data Retrieval (Off-Chain):**
   * Fetch all token-holding wallets from the blockchain.  
     ⬇
2. **Eligibility Filtering (Off-Chain):**
   * Exclude wallets below the minimum balance threshold.
   * Exclude wallets flagged for cooldown.
   * Exclude blacklisted wallets.  
     ⬇
3. **Random Selection (On-Chain):**
   * Send eligible wallets to Chainlink VRF.
   * Receive a random selection of wallets (10% or max 1,000).  
     ⬇
4. **Reward or Burn Execution (On-Chain):**
   * **Reward Day:** Distribute 1% of the reward wallet to selected wallets.
   * **Burn Day:** Burn 1% of the reward wallet.  
     ⬇
5. **Post-Execution Adjustments (Off-Chain):**
   * Update wallet balances.
   * Record transactions on-chain.  
     ⬇  
     **End**

### ****Next Steps****

1. **Finalize Off-Chain Workflow:**
   * Solidify eligibility filtering and wallet data retrieval.
   * Document dependencies and API endpoints for blockchain data access.
2. **Optimize On-Chain Logic:**
   * Minimize gas usage by streamlining on-chain operations.
   * Test Chainlink VRF integration in a simulated environment.
3. **Deploy and Test:**
   * Conduct integration tests for all components.
   * Validate reward and burn cycles using mock wallet datasets.
4. **Prepare Documentation:**
   * Update README and developer guides.
   * Document testing outcomes for future iterations.