**1. Understanding the Script**

This middleware script does the following:

1. **Pulls wallet data from the blockchain** (simulated with an API endpoint).
2. **Processes the data off-chain** to filter eligible wallets.
3. **Uses randomness for wallet selection** (currently simulated but designed to integrate with Chainlink VRF).
4. **Exports the result for use in blockchain interactions** (for reward distribution).

It’s a mix of **on-chain and off-chain operations**.

**2. What You Need to Run This Script**

**Programming Language & Tools**

1. **Python**: The script is written in Python. Download and install it from [python.org](https://www.python.org/downloads/).
2. **Dependencies**: Python libraries like requests, json, and potentially random (all included in Python by default).

**Environment**

1. **Code Editor**: Use an IDE like [VS Code](https://code.visualstudio.com/) or PyCharm.
2. **Terminal/Command Prompt**: To run the script.
3. **API Access**: You’ll need access to the blockchain API endpoint (or simulate one for testing).

**3. Installing Python & Setting Up the Environment**

1. **Install Python**:
   * Download Python from [python.org](https://www.python.org/downloads/).
   * During installation, make sure to check **“Add Python to PATH”**.
   * Verify installation by typing this in your terminal:

bash

CopyEdit

python --version

1. **Install Required Libraries**:
   * Open your terminal or command prompt and run:

bash

CopyEdit

pip install requests

**4. Running the Script**

1. **Save the Script**:
   * Save the Python code into a file named middleware.py.
2. **Execute the Script**:
   * Open a terminal/command prompt, navigate to the folder where middleware.py is saved, and run:

bash

CopyEdit

python middleware.py

1. **View the Outputs**:
   * The script will generate output files like wallet\_snapshot.json and selected\_wallets.json.
   * These files will contain wallet data and selected wallets for rewards.

**5. Dependencies for Production Use**

To make this work in a live environment, you’ll need:

1. **Blockchain API Access**:
   * Replace BLOCKCHAIN\_API\_URL with the actual API endpoint for Shido or your test network.
   * If unavailable, simulate the API response with test data.
2. **Chainlink VRF Integration**:
   * Right now, wallet selection uses Python’s random module as a placeholder.
   * For production, integrate Chainlink VRF to ensure tamper-proof randomness.
3. **Database or Storage**:
   * Use a database (e.g., PostgreSQL, MySQL, or even Firebase) to store and manage wallet snapshots, cooldowns, and other metadata.

**6. Next Steps**

1. **Test the Script Locally**:
   * Simulate API responses by creating a wallet\_snapshot.json file with test data:

json

CopyEdit

[

{"address": "0xWallet1", "balance": 1500, "last\_sale": "2025-01-10", "sale\_percentage": 5},

{"address": "0xWallet2", "balance": 900, "last\_sale": "2025-01-08", "sale\_percentage": 15}

]

* + Modify the script to load this file instead of calling the blockchain API:

python

CopyEdit

def fetch\_wallet\_data():

with open("wallet\_snapshot.json") as f:

return json.load(f)

1. **Collaborate with Your Dev**:
   * Share this script with your developer and discuss how to integrate it into your broader system.
   * They’ll need to set up the **real API access** and **Chainlink VRF integration**.

**Cheat Sheet for Talking to Your Dev**

* **What this script does**: It filters wallets based on eligibility criteria and selects wallets for rewards using randomness.
* **Dependencies**:
  + Blockchain API access (replace BLOCKCHAIN\_API\_URL with the real endpoint).
  + Chainlink VRF for random selection.
  + Python environment with required libraries installed.
* **Current Limitations**:
  + Randomness is simulated with Python’s random module.
  + Wallet data relies on API or simulated JSON.

**7. Hands-On Crash Course**

Here’s how you can practice running the script:

1. **Simulate API Response**: Use the test data example provided above.
2. **Run and Debug**: Follow the steps to execute and check the results.
3. **Ask Questions**: If you hit any roadblocks, let me know—I’m here to guide you.

How’s that for a crash course? Ready to get hands-on, or shall we tackle anything specific? 🚀