Load Balancer

Overview

You are required to redesign the load balancer logic. The task includes two provided code components:

1. Code to generate random IP addresses.

2. Code to determine which node is hit.

These two must be included along with your redesigned load balancer function that replaces the existing random-node routing strategy.

```
// Random IP generator

function generateRandomIP() {

   return Array.from({ length: 4 }, () => Math.floor(Math.random() *
256)).join(".");

}

// List of nodes

const nodes = ["Node-A", "Node-B", "Node-C"];

// Identify which node received the request

function identifyNode(ip, selectedNode) {

   console.log(`Incoming IP: ${ip} → Routed to: ${selectedNode}`);

}

// Temporary Load Balancer function

function LoadBalancer(ip) {

   // Update this Code

   const randomIndex = Math.floor(Math.random() * nodes.length);

   const selectedNode = nodes[randomIndex];


   // Keep this code to identify which node received the request

   identifyNode(ip, selectedNode);

   return selectedNode;

}
```

```
// Simulate incoming traffic
function simulateTraffic(requestCount = 5) {
   for (let i = 0; i < requestCount; i++) {
       const ip = generateRandomIP();
       LoadBalancer(ip);
   }
}
// Run simulation for 10 requests
simulateTraffic(10);
```

Core Features

1. Replace random selection with Proper Required Algorithm for Load Balancing.
2. An IP Should Always reach to the same node even if no. of Nodes are changed or it appears again later.
3. Add logging for each routed request.
4. Optional beginner-friendly enhancements (health check, fallback, etc.).

Constraints

- In-memory structures only
- No concurrency handling required
- Keep logic simple and beginner-friendly

Bonus Challenges

- Basic node health checks
- Weighted routing- To prioritize some Server Nodes
- Simple metrics dashboard
- Rate Limiting Logic

Deliverables

Primary deliverable:
GitHub repository containing the backend project implemented in a suitable framework (for example, Node.js/Express or Python/Flask). The repository should include clear setup and run instructions.

Optional deliverable:
A short demonstration (maximum 2 minutes) in CLI or Postman showcasing the API endpoints and a sample scheduling flow. This demo is optional and may be provided as a short video or a Postman collection with example requests.