Name – harshali deore
Branch – MCA
Email – harshalideore2002@gmail.com

REST API Rate Limiting

Scenario: You are tasked with implementing rate limiting for a REST API endpoint to prevent abuse. The rate limit should be applied per IP address.

Requirements:

- Describe a basic approach to implement rate limiting in a Node.js application.
- What libraries or strategies would you use to handle rate limiting?

--->

1) **Basic approach** –

Tracking Requests: To keep track of how many requests an IP address makes in a certain period of time, keep a data structure (such as a Redis or Map).
Verifying the Rate Limit: Prior to executing a request, ascertain the quantity of requests sent by the present IP address. Provide a response stating that the rate limit has been reached if the limit is surpassed.
Updating Counter: Process the request and update the IP address request counter if it falls under the allowed amount.
Expiration: To avoid memory leaks and eliminate out-of-date records, regularly clean up the data structure.

Using libraries –
Rate restriction in Node.js can be implemented using a variety of libraries and techniques:

Libraries:
Rate Limit: A well-liked and kept library offering a range of solutions for rate limitation (e.g., fixed window, sliding window, token bucket).
Express Rate Limit: An Express.js middleware that makes rate limiting implementation easier.
Redis: A popular in-memory data store for rate limiting that is extremely scalable. To track requests and establish expiration dates, use the incr and expire commands.

- **express-rate-limit:**

  One of the most widely used middleware libraries for rate limitation in Express applications.
  Qualities:
  - Easy setup for time frames, request caps, and personalized error messages. can be used on particular routes or worldwide.

- Tracks requests automatically for each IP and resets the counter after the given window.

- Allows for distributed rate-limiting in bigger, horizontally scaled systems through integration with Redis.

rate-flexible-limiter:

More sophisticated and adaptable rate-limiting library that is not limited to Express and can be used with a variety of Node.js apps.

Qualities:

- Supports distributed or in-memory backends for persistent rate limit tracking, such as Redis, Memcached, and MongoDB.
- Gives the user detailed control over rate-limiting rules, including the ability to establish custom time frames, request caps, and even "burst" processing, which permits brief spikes in request volume.