This is my approach

I built a Node.js API that handles user-specific task processing with rate limiting and queuing. To ensure scalability and performance, I used Node.js clustering to run multiple instances of the API,one for each CPU core, allowing the app to handle more requests concurrently.

For rate limiting, I applied two controls:

1 task per second and20 tasks per minute, both on a per-user basis using express-rate-limit.This ensures each user's requests are tracked and limited independently.Task queuing is managed with Bull and Redis. Bull adds tasks to a queue, and Rediscoordinates the workers to ensure tasks are processed in order and rate limits are respected.If a user exceeds their rate limit, their tasks are queued and processed later. Redis ensures consistency across all worker instances.

I also created a simple logging mechanism that records task completions (including user ID and timestamps) in a log file, so there's a persistent record of all processed tasks.

Finally, I added error handling to ensure the system can recover from failures, like Redis downtime,

without losing tasks.

Instructions on how to run and test My solution.

Install Dependencies:

Run npm install to install all necessary Node.js packages, including Express, Bull, Redis, express-rate-limit.

Start Redis:

Start Redis by running redis-server (or use Docker if you're on Windows). Verify it's working by running

redis-cli ping, and you should get a PONG response.

Run the App:

Start the Node.js app using node app.js for a single instance or use pm2 with the command

pm2 start app.js -i max to run it on multiple CPU cores.

Test the API:

Use Postman, curl, or any HTTP client to send a POST request to http://localhost:3000/api/v1/task

with a JSON body like:

```
{
  "user_id": "123"
}
```

Check Rate Limiting:

Send more than 1 request per second or 20 requests per minute with the same user_id and you should receive a 429 Too Many Requests error.

View Task Logs:

Task completion logs (with user ID and timestamp) are saved in the task_logs.txt file. Check this file to

see processed tasks.