Node.js Task Queuing with Rate Limiting

This project is a Node.js API that manages task queuing and rate limiting for user requests. Each user is limited to 1 task per second and 20 tasks per minute, with tasks queued if the rate limit is exceeded.

Run Locally

- Install dependencies

```
npm install
```

- Start the server

```
node app.js
```

Usage

- Start the Server:
  After running **node app.js**, the server will start with two workers listening on localhost:3000.

- Test the API:
  Send a POST request to **http://localhost:3000/task** with
  JSON body:
  {
  "user_id": "123"
  }
  Used **Postman** for testing

Tech Stack

- **Node.js:** Backend API and server logic
- **Express:** HTTP server and routing
- **Redis:** Task queueing and rate limit management
- **Cluster:** Multi-process setup for load balancing

Approach

- **Rate Limiting**: The application uses Redis to track task counts per user per second and per minute.
- **Task Queueing**: If a user exceeds the rate limit, tasks are queued in memory and executed when within limits.
- **Logging**: Each completed task logs the user ID and timestamp in task_log.txt.
- **Clustering**: The application uses Node's cluster module to spawn two workers, enhancing performance and reliability.

Assumptions

- Each user has a unique user_id, used for tracking rate limits.
- Redis is configured to run on localhost:6379.
- Rate limits apply independently per user.

Preservation of Requests

- The application guarantees that **no requests are dropped**. If requests exceed the rate limit, they are preserved in the queue and processed as soon as permitted by the rate limits. This ensures that every task is eventually completed in sequence.