Node Assignment: User Task Queuing with Rate Limiting

Task:

Your task is to build a Node.js API cluster with two replica sets and create a route to handle a simple task. The task has a rate limit of 1 task per second and 20 task per min for each user ID. Users will hit the route to process tasks multiple times. You need to implement a queueing system to ensure that tasks are processed according to the rate limit for each user ID.

Task Function:

You are provided with the following task function:

```
async function task(user_id){
    console.log(`${user_id}-task completed at-${Date.now()}`)
// this should be stored in a log file
}
```

This function logs the completion of a task along with the user ID and the timestamp. You are required to store this information in a log file.

Request Structure

- **Route:** `/api/v1/task`
- **Method:** POST
- **Body:** JSON

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

Rate Limiting:

- The rate limit for each user ID is one task per second and 20 task per minute.
- Requests exceeding the rate limit should be queued and processed accordingly.
- Ensure that the rate limiting mechanism is user-based.

Requirements:

1. Set up a Node.js API cluster with two replica sets.
2. Implement rate limiting to enforce one task per second per user ID.
3. Create a task queueing system to manage tasks for each user ID.
4. Implement the provided task function to log task completion along with user ID and timestamp.
5. Store task completion logs in a log file.
6. Ensure that the API is resilient to failures and edge cases.

Submission:

Submit your solution as a compressed archive containing all necessary files, including source code, configuration files, and documentation explaining your approach and any assumptions made.

Evaluation:

Your submission will be evaluated based on the following criteria:

- Correct implementation of rate limiting and task queueing mechanisms.
- Proper handling of asynchronous operations and edge cases.
- Efficiency and scalability of the solution.
- Clarity and organization of the codebase.
- Documentation quality and adherence to best practices.

Additional Notes:

- You may use any libraries or frameworks you deem appropriate for the task.
- Redis is recommended for queueing between clusters.
- Ensure that your solution is well-documented and easy to understand.
- Provide clear instructions on how to run and test your solution.
- Avoid hardcoding sensitive information such as file paths or API keys.
- Test your solution thoroughly to ensure correctness and reliability.

Good luck with your assignment!