Backend Developer - Sample Task (MathonGo)

Objective

Build a RESTful API-based backend for a **Chapter Performance Dashboard**. The goal is to simulate real-world backend requirements such as API design, data filtering, caching, and performance optimization.

Resources

Mock JSON Data: Link

Requirements

Tech Stack

- Node.js
- Express.js
- MongoDB (with mongoose)
- Redis (for caching & rate-limiting)

Features to Implement

1. **RESTful API Endpoints**
    - Implement following endpoints

```
GET    /api/v1/chapters (returns all the chapters)
GET    /api/v1/chapters/:id (returns a specific chapter)
POST   /api/v1/chapters (upload chapters to DB)
```

    - Chapter uploading should only be allowed to admin. One should be able to upload a JSON file and the backend needs to parse the file in order to upload chapters. If any error is found in one of the chapters during schema validation, add the rest of the chapters and finally return the chapters that failed uploading.
    - For getting all the chapters, following filters should be allowed to the client:
        i. `class`, `unit`, `status`, `weakChapters` and `subject`
        ii. `page` and `limit` should be received in order to paginate the results, requests should also send the total number of chapters present
2. **REDIS Caching**
    - Cache the `/api/v1/chapters` endpoint results in Redis for 1 hour

    - Invalidate the cache when a new chapter is added
3. **Rate Limiting**
    - Only allow 30 requests/minute per IP Address. Make sure to use Redis in order to rate limit your routes

Submission Instructions

1. Create a public GitHub repository with a random name (to avoid plagiarism by other candidates) and push the code and copy the repo link.
2. Create a public postman collection containing all the routes in a well documented way and pre populated data
3. **Deploy the app** on **Render**, **Railway**, **Fly.io**, or any other free-tier backend hosting service. **Bonus if deployed on an EC2 instance with a github workflow file**
4. **Submit the Google Form:** Link

Note: Feel free to use any AI tools which can improve your workflow and efficiency. We'll not discard if we find a `.cursorrules` file in the repo. So, yes, Cursor, Claude, ChatGPT, anything is fine as long as you know what you're doing!

Scoring Breakdown (100 Marks)

Section	Criteria	Marks
API Functionality	Working endpoints with correct filtering & responses	30
Caching (Redis)	Proper cache setup, invalidation logic	15
Rate Limiting (Redis)	Effective Rate Limiting of requests based on IP Address	10
Pagination	Client based query param logic returning the accurate requested data	10
Code Quality	Clean, modular and error-handled code	15
Deployment	Hosted API with working endpoints	10
Bonus	Deployment on EC2 along with a workflow file	10

Timeline to Submit

The submission should be done **within 3 days of receipt** of this sample task.