Retail Pulse Assignment

❑ Description:

- This project implements a backend service to process thousands of images collected from retail stores. The service supports submitting jobs to calculate image perimeters, simulating GPU processing through randomized delays, and storing results. Additionally, it allows users to query job status and handle errors for invalid image URLs or store IDs.

❑ Assumptions you took if any:

- Image Dimensions: Image dimensions (Height and Width) can be derived using standard libraries like image in Go after downloading.
- Random Sleep for GPU Simulation: The random sleep time between 0.1 and 0.4 seconds is uniform and simulates GPU load during image processing.
- Job Persistence: For simplicity, job data is stored in-memory (e.g., sync.Map). In production, a database like PostgreSQL or MongoDB would be used.
- API Request Validation: Basic validation is done on fields like store_id, image_url, and count.
- Store Master Data: Assumes the store master file (Store_Master.csv) is accessible in the project directory and loaded during server startup.

❑ Installing (setup) and testing instructions:

- Clone the Repository:

bash

git clone https://github.com/SoumyaBehura18/backend-intern-assignment

cd backend-intern-assignment

- Set Up Dependencies:

Ensure you have [Go](https://golang.org/dl/) installed (v1.20 or higher). Run:

bash

go mod tidy

- Run Without Docker:

Start the server directly:

bash

```
go run main.go
```

Test the APIs using Postman, cURL, or any API client:

- ✓ Submit Job: POST http://localhost:8080/api/submit
- ✓ Get Job Info: GET http://localhost:8080/api/status?jobid=<job_id>
- Run Using Docker:
- ✓ Build the Docker image:

bash

```
docker build -t backend-assignment .
```

- ✓ Run the container:

bash

```
docker run -p 8080:8080 backend-assignment
```

- Testing:
- ✓ Submit a sample job:

bash

```
curl -X POST http://localhost:8080/api/submit -H "Content-Type: application/json" -d
'{
  "count": 1,
  "visits": [
    {
      "store_id": "S00339218",
      "image_url": [
        "https://www.gstatic.com/webp/gallery/2.jpg"
      ],
      "visit_time": "2024-11-17T10:00:00Z"
    }
```

```
 ]
}'
```

- ✓ Query the job status:

bash

```
curl -X GET "http://localhost:8080/api/status?jobid=<job_id>"
```

❑ Brief description of the work environment used to run this project (Computer/operating system, text editor/IDE, libraries, etc):

- Operating System: Windows 11 Pro
- IDE/Editor: Visual Studio Code with Go plugins
- Libraries:
  - ✓ gorilla/mux for HTTP routing
  - ✓ sync for concurrency management
  - ✓ image and image/jpeg for image processing
- Development Tools:
  - ✓ Docker Desktop for containerization
  - ✓ Postman for API testing

❑ If given more time, what improvements will you do?:

- 1. Database Integration: Use PostgreSQL or MongoDB for persistent job storage.
- Queue System: Integrate a job queue like RabbitMQ or Kafka to handle multiple jobs more efficiently.
- Error Handling: Enhance error logging and reporting for debugging and monitoring.
- Scalability: Implement horizontal scaling with Kubernetes or Docker Swarm.
- Authentication: Add token-based authentication (e.g., JWT) for secure API access.
- Improved Image Processing: Optimize the image download and processing pipeline with parallelization.