Throttling Engine for Priority-Based API Requests

You're tasked with building a **throttling engine** for a high-traffic API platform where users fall into different **priority groups**:

- **normal**
- **premium**
- **VIP**

Objectives

1. **Separate queues per priority group**
    - Maintain individual request queues for each group: "normal", "premium", "vip"
2. **Global processing limit**
    - A maximum of N requests can be processed **concurrently** across all groups (e.g., N = 10)
3. **Fairness**
    - Ensure **fair distribution** of processing slots using **round-robin** or **weighted scheduling**
4. **Request simulation**
    - Each request includes userID, group, payload, and simulated duration (e.g., 500ms–1500ms)

---

1. Request Submission

Clients submit requests like:

```
Request{
  UserID: "user-123",
  Group:  "premium",
  Payload: "do-something",
  Duration: 1200 * time.Millisecond
}
```

---

2. Request Queues

Each group has:

- Its **own FIFO queue**

- A **quota** for how many requests can be processed in a scheduling cycle:
    - "normal": 2
    - "premium": 3
    - "vip": 5

---

3. Scheduler

A **weighted round-robin scheduler**:

- Pulls up to each group's **quota** worth of requests per round
- Respects the global concurrency limit
- Simulates processing by sleeping (time.Sleep(duration))

📌

Sample Flow

Assume the following conditions:

- Global max concurrent requests: 10
- Group quotas: normal=2, premium=3, vip=5

Requests Arrive:

Time (ms)	User ID	Group	Duration
0	U1	normal	800ms
10	U2	vip	1000ms
20	U3	vip	1200ms
30	U4	premium	900ms
40	U5	vip	600ms
50	U6	normal	1000ms
60	U7	premium	1100ms
70	U8	vip	700ms
80	U9	premium	950ms

90 U10 vip 800ms

Scheduler Round:

- Picks up to 5 from VIP: U2, U3, U5, U8, U10
- Picks up to 3 from Premium: U4, U7, U9
- Picks up to 2 from Normal: U1, U6

➡️ 10 total concurrent requests picked, respecting group quotas and global max.

V - 40 -35-30-25-20-15-10-5-0
P -30 - 27-24-21-18-15-12-9-6-3-0
N -30-28 - 26

5
3
2