package main

import (

"net/http"

"sync"

"time"

)

type rateLimiter struct {

requests map[string]int

mu sync.Mutex

limit int

window time.Duration

}

func newRateLimiter(limit int, window time.Duration) \*rateLimiter {

return &rateLimiter{

requests: make(map[string]int),

limit: limit,

window: window,

}

}

func (rl \*rateLimiter) allow(ip string) bool {

rl.mu.Lock()

defer rl.mu.Unlock()

if rl.requests[ip] >= rl.limit {

return false

}

rl.requests[ip]++

go func() {

time.Sleep(rl.window)

rl.mu.Lock()

rl.requests[ip]--

rl.mu.Unlock()

}()

return true

}

func rateLimitingMiddleware(rl \*rateLimiter) func(http.Handler) http.Handler {

return func(next http.Handler) http.Handler {

return http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

clientIP := r.RemoteAddr

if !rl.allow(clientIP) {

http.Error(w, "Too Many Requests", http.StatusTooManyRequests)

return

}

next.ServeHTTP(w, r)

})

}

}

func main() {

rl := newRateLimiter(5, 10\*time.Second) // Limit: 5 requests per 10 seconds

http.Handle("/rate-limited-endpoint", rateLimitingMiddleware(rl)(http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

w.Write([]byte("Request Successful!"))

})))

http.ListenAndServe(":8080", nil)

}