SEND THE CODE 2 OURS PRIOR TO INTERVIEW FOR TAKE-HOME (SEPARATELY)

# **Overview**

In this exercise, you will implement a rate limiting algorithm to control the number of requests a customer can make over a specified time period. You may choose any rate limiting algorithm you prefer (fixed window, sliding window, token bucket, leaky bucket, etc.).

## **Requirements**

Implement a rate limiter with the following interface:

// Class that implements rate limiting functionality  
class RateLimiter:  
  
 // Constructor that initializes the rate limiter  
 // rate: maximum number of requests allowed per time window  
 // time\_window: the time window in seconds over which to apply the rate limit  
 constructor(rate, time\_window)  
  
 // Method that determines if a request should be allowed or rejected  
 // customer\_id: unique identifier for the customer making the request  
 // current\_time: the current timestamp in seconds since the epoch  
 // Returns: true if the request is allowed, false if it should be rejected  
 is\_allowed(customer\_id, current\_time) -> boolean

## **Expectations**

1. Your solution should correctly limit requests according to the specified rate and time window
2. The implementation should handle multiple customers, each with their own rate limit
3. Your code should be well-structured, readable, and include appropriate comments
4. Include basic error handling for edge cases
5. You may use any data structures needed to efficiently implement your chosen algorithm

## **Example Usage**

// Example using a fixed window algorithm:  
rate\_limiter = new RateLimiter(5, 60) // 5 requests per 60 seconds  
  
// First 5 requests from customer\_id "user\_123" at time 100 seconds  
is\_allowed("user\_123", 100) // Returns true  
is\_allowed("user\_123", 110) // Returns true  
is\_allowed("user\_123", 115) // Returns true  
is\_allowed("user\_123", 120) // Returns true  
is\_allowed("user\_123", 125) // Returns true  
  
// 6th request from customer\_id "user\_123" at time 130 seconds (still within window)  
is\_allowed("user\_123", 130) // Returns false  
  
// Request from customer\_id "user\_123" at time 170 seconds (in a new window)  
is\_allowed("user\_123", 170) // Returns true

## **Additional Considerations**

* Think about the time and space complexity of your implementation
* Consider how your solution might scale with a large number of customers

## **Testing**

Include tests that demonstrate your rate limiter correctly handles various scenarios, including:

Multiple requests within and across time windows Multiple customers making requests simultaneously Edge cases such as time rollovers or resets

Please implement this solution in your preferred programming language. If you choose the take-home option, include a README explaining your approach and any assumptions you made.