**Semaphore – API**

'Should acquire a lock if no blocking semaphores exist.'

'Should reject when acquiring and another exclusive semaphore exist for different process key.'

'Should allow multiple shared semaphores when limit is not exceeded.'

'Should reject acquiring shared semaphore when concurrent Process Limit reached.'

'Should reject acquiring shared semaphore when setting limit below current number of semaphores and total number of processes exceed it'

'Should allow downsizing concurrent Process Limit when total number of processes doesn’t exceed it.'

'Should allow multiple shared semaphores without any limit.'

'Should reject exclusive semaphore when shared exist and process Run Keys are different.'

'Should acquire exclusive semaphore when shared exist and process Run Keys are the same.'

'Should acquire multiple locks for exclusive semaphore for the same process key.'

Archive

should remove semaphores older than provided retention time.

Fetch

should fetch semaphores.

Release

should release orphaned semaphores.

should release semaphore.

should return conflict when releasing non-existent semaphore.

Renew

should renew lock.

should return 404 when renewing non-existent semaphore.

**I. Feature: Semaphore Locking**

**1**. **Scenario: Acquire Lock When No Blocking Semaphores Exist**

Given there are no blocking semaphores in the system.

When a lock is requested

Then the lock should be acquired successfully

And the system should indicate that the lock has been acquired.

**2.Scenario: Reject Acquiring Lock When Another Exclusive Semaphore Exists for Different Process Key**

Given there is an exclusive semaphore held by a different process key.

When a lock is requested for a different process key

Then the lock acquisition should be rejected

And the system should indicate that the lock acquisition has failed due to an existing exclusive semaphore.

**3.Scenario: Allow Multiple Shared Semaphores When Limit is Not Exceeded**

Given the system allows for multiple shared semaphores.

And the current number of shared semaphores does not exceed the limit.

When multiple shared semaphores are requested

Then the system should allow the acquisition of each semaphore

And indicate that each semaphore has been acquired successfully.

**4**. **Scenario: Reject Acquiring Shared Semaphore When Concurrent Process Limit Reached**

Given the system has reached the concurrent process limit for shared semaphores.

When a shared semaphore is requested

Then the system should reject the acquisition of the semaphore

And indicate that the acquisition has failed due to reaching the concurrent process limit.

**5. Scenario: Reject Acquiring Shared Semaphore When Limit is Set Below Current Semaphore Count and Total Processes Exceed It**

Given the system has more shared semaphores currently acquired than the new limit.

And the total number of processes exceeds the new limit.

When a shared semaphore is requested

Then the system should reject the acquisition of the semaphore

And indicate that the acquisition has failed due to exceeding the new limit.

**6. Scenario: Allow Downsizing Concurrent Process Limit When Total Number of Processes Doesn't Exceed It.**

Given the system has a current concurrent process limit.

And the total number of processes does not exceed the current limit.

When the concurrent process limit is downsized

Then the system should allow the limit to be changed successfully

And indicate that the new limit has been applied.

**7. Scenario: Allow Multiple Shared Semaphores Without Limit**

Given there is no limit set for the number of shared semaphores.

When multiple shared semaphores are requested

Then the system should allow the acquisition of each semaphore

And indicate that each semaphore has been acquired successfully.

**8. Scenario: Reject Exclusive Semaphore When Shared Semaphore Exists and Process Run Keys Are Different**

Given there is a shared semaphore held by a different process run key.

And the process run keys for the shared semaphore and the requested exclusive semaphore are different.

When an exclusive semaphore is requested

Then the system should reject the acquisition of the exclusive semaphore

And indicate that the acquisition has failed due to the presence of a shared semaphore with a different process run key.

**9. Scenario: Acquire Exclusive Semaphore When Shared Semaphore Exists and Process Run Keys Are the Same**

Given there is a shared semaphore held by the same process run key.

And the process run keys for the shared semaphore and the requested exclusive semaphore are the same.

When an exclusive semaphore is requested

Then the system should allow the acquisition of the exclusive semaphore

And indicate that the exclusive semaphore has been acquired successfully.

**10. Scenario: Acquire Multiple Locks for Exclusive Semaphore for the Same Process Key**

Given there is an exclusive semaphore held by a process key.

When multiple locks are requested for the exclusive semaphore with the same process key

Then the system should allow the acquisition of each lock

And indicate that each lock for the exclusive semaphore has been acquired successfully.

**II. Feature: Semaphore Lock Release**

1.**Scenario: Remove Semaphores Older Than Provided Retention Time**

Given there are semaphores in the system.

And some of these semaphores are older than the provided retention time.

When the cleanup process is triggered

Then the system should remove semaphores older than the provided retention time

And indicate that the cleanup process has been successfully completed.

**2. Scenario: Release Orphaned Semaphores**

Given there are orphaned semaphores in the system.

When the release orphaned semaphores process is triggered

Then the system should release all orphaned semaphores

And indicate that the release process was successful.

**3.Scenario: Fetch Semaphores**

Given there are semaphores stored in the system.

When the fetch semaphores operation is performed

Then the system should return a list of all semaphores

And indicate that the operation was successful.

**4. Scenario: Release Semaphore**

Given there is a semaphore currently held in the system.

When the release semaphore operation is performed

Then the system should release the semaphore

And indicate that the semaphore has been successfully released.

**5. Scenario: Return Conflict When Releasing Non-Existent Semaphore**

Given there is no semaphore currently held in the system.

When an attempt is made to release a semaphore

Then the system should return a conflict status

And indicate that the semaphore does not exist.

**IV. Feature: Semaphore Lock Renewal**

**1.Scenario: Renew Lock**

Given there is an existing lock held in the system.

When the renew lock operation is performed

Then the system should renew the lock

And indicate that the lock has been successfully renewed.

**2.** **Scenario: Return 404 When Renewing Non-Existent Semaphore**

Given there is no semaphore currently held in the system.

When an attempt is made to renew the semaphore

Then the system should return a 404 status

And indicate that the semaphore does not exist.