**Test Results**Team Decided - Raft Consensus Library

### Test 1 - Can Read/Write to Debug Log

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Reading Log | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | The library writes out to a log file for each of it’s noteworthy operations, it also allows multiple log levels and the logs show a clear description of what is going on within the library | | | | | | |
| **Pre-Conditions:** | | At least 1 node has been instantiated | | | | | | |
| **Post-Conditions:** | | Logs have been created and is readable by a developer | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Adds the required lines for testing, specifying log level | | | Code is available, understandable, and compiles | N/A |  | Pass | |
| 2 | Runs at least 1 node | | | Node starts up and starts writing to log | N/A |  | Pass | |
| 3 | Confirms debug file created, and shows clear description of what is going on | | | Log is created and contains information relating to state | Show debug available to UAS, and debug file | 1.1, 1.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 1.1 | | |  | | | | | |
| 1.2 | | |  | | | | | |

### Test 2 - Can Read/Write to Distributed Log

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Append Entry | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | Active UAS receives append entry message and writes data to distributed log | | | | | | |
| **Pre-Conditions:** | | The current node should be running a UAS | | | | | | |
| **Post-Conditions:** | | An entry is entered into the distributed log | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Start cluster | | | Cluster starts up | N/A |  | Pass | |
| 2 | Append entry to cluster | | | Can enter data into Leader | N/A |  | Pass | |
| 3 | Verify it appears in other nodes log | | | Entry is available in log | Image of data propagated successfully across nodes, showing reading capability | 2.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 2.1 | | |  | | | | | |

### Test 3 - Can Send and Receive Messages From Other Nodes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | All use cases involving sending messages | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | Verifying network connectivity, from confirming Test 2 has a successful result we can confirm that messages are being sent across the network between nodes | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | | Test 2 has been run | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 2 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 4 - Can Communicate Using Encrypted Messages

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | All use cases involving sending messages | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Verify that the data is encrypted when selecting to use encryption for the network communication, and unencrypted otherwise | | | | | | |
| **Pre-Conditions:** | | A cluster started, sending messages between each other | | | | | | |
| **Post-Conditions:** | | Cluster has sent messages | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Run the cluster and commit an entry with encryption disabled | | | Entries commit | N/A |  | Pass | |
| 2 | Use packet analysis software to view the body of the message at the network layer | | | Can see plain text data | Image of plain text data | 4.1 | Pass | |
| 3 | Run the cluster and commit an entry with encryption enabled | | | Entries commit | N/A |  | Pass | |
| 4 | Use packet analysis software to view the body of the message at the network layer | | | Can see plain text data is encrypted | Image of encrypted data | 4.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 4.1 | | |  | | | | | |
| 4.2 | | |  | | | | | |

### Test 5 - Node authenticates using zero knowledge password proof

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Join Cluster | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Developer displaying sequence diagram of encryption handshake, and showing in code where this is implemented. | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer showing in code where the zero knowledge proof challenge response code is written | | | Code displays extract where zero knowledge proof is conducted | Viewing code extract | 5.1 | Pass | |
| 2 | Developer showing sequence diagram for handshake, showing challenges/response | | | Sequence diagram shows handshake and encryption protocol | Viewing sequence diagram | 5.2 |  | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 5.1 | | |  | | | | | |
| 5.2 | | |  | | | | | |

### Test 6 - Does Library Call UAS Start/Stop

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Start UAS/Stop UAS | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | Showing UAS starting/stopping when becoming leader of the cluster | | | | | | |
| **Pre-Conditions:** | | Have a running cluster | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Become the leader | | | Node becomes leader | N/A |  | Pass | |
| 2 | Show UAS starts | | | Textboxes become able to write in  UAS becomes “Active” | Show UAS starts on Node1, textbox’s are able to be written into, and “Active” is shown. This shows the Start UAS event was raised when gained leadership | 6.1 | Pass | |
| 3 | Lose leadership | | | Leadership lost | N/A |  | Pass | |
| 4 | Show UAS stops | | | Textboxes on old leader become unavailable  UAS stops being “Active” | Node1 loses it’s UAS, so it’s textbox’s become unavailable, state is no longer “Active” | 6.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 6.1 | | |  | | | | | |
| 6.2 | | |  | | | | | |

### Test 7 - Can Hold Successful Election

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of verifying underlying code functionality | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | This is a test on the underlying code functionality of the library, and its ability to hold successful elections. The winner of these elections becomes leader, and is able to resume running the UAS. | | | | | | |
| **Pre-Conditions:** | | Cluster has started, no leader yet | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Node times out and requests to be leader | | | Node becomes candidate | Node debug log showing candidate mode and getting votes | 7.1 | Pass | |
| 2 | Node receives successful votes back from other nodes and becomes leader, starts UAS | | | Node gets elected | Node debug log changing to leader | 7.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 7.1 | | |  | | | | | |
| 7.2 | | |  | | | | | |

### Test 8 - Bring Node Log Up To Date

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Receive Notice of Entry Commit | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | When a node restarts after a crash, the leader will bring it up to date with the rest of the cluster | | | | | | |
| **Pre-Conditions:** | | Cluster available, entries have been committed, one node has crashed | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Node starts back up, becomes follower in cluster | | | Starts up with UAS not Active | Restated node has no entries while the rest of the cluster has entries | 8.1 | Pass | |
| 2 | Node gets it’s log rebuilt by the leader | | | Displays the log entries | Node new up to date with leader again | 8.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 8.1 | | |  | | | | | |
| 8.2 | | |  | | | | | |

### Test 9 - Falls to Follower When Detecting Newer Leader

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Fault tolerant distributed service” functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Developer displaying that a node in leader would fall to follower if receives message from a leader with a newer term | | | | | | |
| **Pre-Conditions:** | | Cluster started, node is leader, node’s network connection to the rest of the nodes is broken, the rest of the cluster detects this and moves on electing a new leader, the old leader’s network connection resumes | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developers shows code extract which is run during this situation | | | Code shows that leader would fall to follower when detecting newer leader | Code showing that leader would fall to follower when detecting newer leader | 9.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 9.1 | | |  | | | | | |

### Test 10 - Can Maintain Service During Node Failure/Loss

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Fault tolerant distributed service” functional requirement | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | Prototype shows that after a **follower** node is lost, can still commit entries | | | | | | |
| **Pre-Conditions:** | | Cluster exists, entries in the log, follower node is stopped | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Leader append new entry to the log | | | Can see entry is propagated out and committed by remaining follower, commit occurs and shows in log | Image of stopped follower, image of new follower having commit the entry for the leader |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 10.1 | | |  | | | | | |

### Test 11 - Can Recover From Node Failure/Loss

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Fault tolerant distributed service” functional requirement | | | | | | |
| **Test Type** | | Prototype | | | | | | |
| **Test Description** | | Prototype shows that after a **leader** node is lost, new leader is found and can still commit entries | | | | | | |
| **Pre-Conditions:** | | Cluster exists, entries in the log | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Disable leader | | | Leader stops UAS and is disabled, can see new leader has been elected | Show old leader turned off, show new leader that is elected | 11.1 | Pass | |
| 2 | Commit entries into new leader | | | New leader can commit entries | Can see new entries are committed by leader | 11.2 |  | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 11.1 | | |  | | | | | |
| 11.2 | | |  | | | | | |

### Test 12 - Consensus Between Distributed Systems

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Fault tolerant distributed service” functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | From confirming Test 2 has a successful result we can also confirm that nodes are able to reach consensus between themselves and show entries in a log | | | | | | |
| **Pre-Conditions:** | | Test 2 has been run | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 2 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 13 - Fault Tolerant Distributed Service

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Fault tolerant distributed service” functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | From confirming Test 2, Test 10 and Test 11 have successful results we can also confirm that nodes are able maintain a fault tolerant distributed service | | | | | | |
| **Pre-Conditions:** | | Test 2 has been run, Test 10 has been run, Test 11 has been run | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 2 is a pass | | | Pass | N/A |  | Pass | |
| 2 | Confirm Test 10 is a pass | | | Pass | N/A |  | Pass | |
| 3 | Confirm Test 11 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 14 - Security - All comms encrypted

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Security” non-functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | From confirming Test 4 and Test 5 have successful results we can also confirm that nodes can communicate security | | | | | | |
| **Pre-Conditions:** | | Test 4 has been run, Test 5 has been run | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 4 is a pass | | | Pass | N/A |  | Pass | |
| 2 | Confirm Test 5 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 15 - Privacy - Joining Securely To Cluster

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Privacy” non-functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | From confirming Test 14 has successful results we can also confirm that nodes can communicate with privacy, as they can security join a cluster. | | | | | | |
| **Pre-Conditions:** | | Test 14 has been run | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 14 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 16 - Cross Platform - Dev. language

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Cross Platform” Functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Shows that chosen language to implement code in is executable cross platform (Windows, Linux, Mobile, etc.) | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Show that code is written in .NET Standard | | | Code is shown to be written in .NET Standard | Image of properties of Raft Consensus Project which shows targeting .NET Standard framework | 16.1 | Pass | |
| 2 | Show that .NET Standard is cross platform | | | .NET standard is shown to be cross platform | Image of website showing .NET standard is cross platform | 16.2 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 16.1 | | |  | | | | | |
| 16.2 | | |  | | | | | |

### 

### Test 17 - Mitigate project abandonment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Mitigate project abandonment” Functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Developer shows that code is open source, and allows for profit | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer shows project is open source | | | Developer can show online version control repo with Bitbucket, however it’s still marked as private until the Final. | Image of repo on Bitbucket | 17.1 | Pass | |
| 2 | Developer shows code library license is “Apache 2.0” | | | Library license is “Apache 2.0” | Library license being in repo is “Apache 2.0” | 17.2 | Pass | |
| 3 | Developer show license “Apache 2.0” allows for profit | | | License “Apache 2.0” shown to allow profit | Snip of “” allowing profit | 17.3 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 17.1 | | |  | | | | | |
| 17.2 | | |  | | | | | |
| 17.3 | | |  | | | | | |

### Test 18 - Compatibility - Dev. language popularity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Compatibility” Functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Developer shows that language code is written is is a popular **application** language. | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer shows proof that language is high up on the list of the most popular languages | | | Language is shown to be popular | Image showing the language is popular | 18.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 18.1 | | |  | | | | | |

### Test 19 - Troubleshooting - Logging

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Reading Log - The developer’s use case | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | This test shows that the running program outputs helpful logs for the developer. It shows multiple possible log levels, and the log produced is reasonably verbose as described by level | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer shows that there are multiple log levels possible | | | Multiple log levels available | Image of log levels from prototype | 19.1 | Pass | |
| 2 | Developer shows that a “Info” log level doesn’t show unnecessary messages | | | Log is simple to read, does not show messages, just state changes | Snip of prototype showing a log which only shows messages about state | 19.2 | Pass | |
| 3 | Developer shows that a “debug” log level provides additional information for debugging issues | | | Log is quiet more verbose, showing messages being sent back and forth | Snip of prototype showing a log which has info about messages being sent/received | 19.3 | Pass | |
| 4 | Developer shows that a “Trace” log level provides even more additional information for debugging issues, allowing them to trace the execution | | | Log is very verbose, showing messages being sent back and forth, also showing inter-method tracing log entries | Snip of prototype showing a log which has info about messages being sent/received and inter-method tracing log entries | 19.4 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 19.1 | | |  | 19.2 |  | | | |
| 19.3 | | |  | 19.4 |  | | | |

### Test 20 - Reliability

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Reliability” functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | From confirming Test 13 is pass, can confirm that the system is a fault tolerant system which enables higher reliability/available of a service during failure scenarios | | | | | | |
| **Pre-Conditions:** | | Test 13 has been run | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Confirm Test 13 is a pass | | | Pass | N/A |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |

### Test 21 - Minimal overhead/impact to service performance

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Minimal overhead/impact to service performance” functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | During distributed consensus, the network is always going to be the slowest point. This part of the communication takes 10s of milliseconds, or more, per entry. This test shows that consensus algorithm implemented is as efficient with it’s message as other implementations. | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer can show evidence that algorithm is as efficient as competing consensus algorithms | | | The Raft consensus algorithm Paper is able to provide this evidence | An image of where in the Raft Paper this is stated | 21.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 21.1 | | | Excerpt from [Raft Paper](https://raft.github.io/raft.pdf) | | | | | |

### Test 22 - Usability - Minimalistic public interface

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Usability” non-functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | The public interface to the raft algorithm should hide all complexity and allow the lowest possible barrier for entry for develops to implement it into their program | | | | | | |
| **Pre-Conditions:** | |  | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer can show that the list of methods available to the user is as few as possible to still be able to conduct consensus | | | Developer is able to show an image displaying this property | Image of the class diagram which shows the methods which are available to the developer | 22.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 22.1 | | |  | | | | | |

### Test 23 - Availability - Can run locally and over internet

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of project “Availability” non-functional requirement | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | Test shows that the algorithm is able to provide it’s increase in availability to a service over an Internet connection | | | | | | |
| **Pre-Conditions:** | | Ports are forwarded in NAT tables on both sides, firewalls allow packets through on both sides | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Forward ports through NAT tables on both sides, forward packets through firewalls on both sides | | | Ports are added to NAT table, packets are allowed through firewall on each side | N/A |  | Pass | |
| 2 | Prototype is able to run across the internet | | | Can show prototype able to communicate between nodes across the internet, and commit entries | Image of prototype running across the internet, and able to commit entries. Remote Desktop software is used to show evidence of other side of connection. |  | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 23.2 | | |  | | | | | |

### Test 24 - Confirm library is one-click integratable from Nuget

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | N/A - Part of Project Vision - Usability (non-functional requirements) | | | | | | |
| **Test Type** | | Developer Evidence | | | | | | |
| **Test Description** | | This test shows that the code is readily available for developers to pull down from Nuget, an open source .NET package manager backed by Microsoft. | | | | | | |
| **Pre-Conditions:** | | Library is pushed up to Nuget and is made available for download | | | | | | |
| **Post-Conditions:** | |  | | | | | | |
| **Notes:** | | [NuGet Package TeamDecided.RaftConsensus](https://www.nuget.org/packages/TeamDecided.RaftConsensus/) | | | | | | |
| **Results** | | Pass | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Evidence Description** | **Evidence ID(s)** | **Result** | |
| 1 | Developer Package shown on NuGet site ready for download | | | Can see the library available for download on the Nuget site | Image of the Nuget site showing the library available | 24.1 | Pass | |
| **Evidence ID** | | | **Evidence** | | | | | |
| 24.1 | | |  | | | | | |