## User Acceptance Beta Tests

For beta testing of our library it was unreasonable to get developers of mission critical software to integrate our library into their own projects as a separate solution for their consensus needs when the code base in only entering the beta stage. So upon discussion with our lecturer we agreed that the best way to demonstrate beta level testing would be to implement it into our own demonstration program. For that, we’ve continued maintaining and adding our newer features into our Prototype program which utilises Microsoft’s Winforms and the .NET 4.6 framework to provide the user a graphical interface to a basic text distributed key/value store. This program can be used to enable users to spin up and test nodes on their local computer, as well as used to create nodes which can talk across the internet.

The following UATs for beta match up with our library’s Use Cases, and we’ve furthermore extended them in cases such as “survive node failure” and “Node Rejoin Cluster/Rebuild” which is typically hidden from the user.

* Install prototype
* Create config
  + Without encryption, without persistent storage
  + With encryption
  + With persistent storage
  + With encryption and persistent storage
* Join Cluster
  + Load node following config creation
  + Load from persistent storage
* Append Entry
* Receive Commit Entries
* Stop Node
* Start Node
* Survive node failure
* Node rejoin cluster/rebuild
* Developer Read Log

### Beta Test 1 - Install Prototype

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Install Prototype | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | |  | | | | | | |
| **Pre-Conditions:** | | User has not yet downloaded or installed Raft Prototype application | | | | | | |
| **Post-Conditions:** | | Raft Prototype is downloaded and installed | | | | | | |
| **Notes:** | | The user is expected to understand how to do the basic task of running through an installer clicking “Next”, and accepting any security dialog popups. | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Download Raft Prototype from [here](https://bitbucket.org/teamdecided/raftprototype/raw/5f16d7772b653c86a59c620c5e18ac51888ece86/RaftPrototypeInstaller/Release/RaftPrototypeInstaller.msi) | | | User download .msi file |  | | | |
| 2 | Proceed through the standard steps of the installer. Accept any warning. | | | User installs software to computer. |  | | | |

### 

### Beta Test 2 - Create Config - Unencrypted/ephemeral storage

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Create Config - Unencrypted/ephemeral storage | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This is the most basic config creation test, it ensures a user is able to build a config describing an unencrypted and ephemeral cluster | | | | | | |
| **Pre-Conditions:** | | User has installed Raft Prototype | | | | | | |
| **Post-Conditions:** | | A Raft Consensus Config “.rcc” file is created, user is left on dialog asking if they’d like to start one of the nodes | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Open Raft Prototype from the shortcut made on the desktop | | | Raft Prototype program opens on computer |  | | | |
| 2 | Select ‘Create cluster config’ | | | Takes user to Create Cluster Config page |  | | | |
| 3 | Use the example configuration of:   * Cluster name - “My first cluster” * Encryption - Leave unchecked * Join Retry Attempts - 5 * Persistent Storage - Leave unchecked * Number of nodes - 3 | | | User can change name  Retry time auto calculates to 50s |  | | | |
| 4 | Select to ‘Build’ the config, and select where to save the file | | | Config builds successfully  Opens dialog window  File is saved to targeted location |  | | | |

### Beta Test 3 - Create Config - Network encryption enabled

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Create Config - Network encryption enabled | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This is config creation test, it ensures a user is able to build a config describing a config with network encryption enabled | | | | | | |
| **Pre-Conditions:** | | User has installed Raft Prototype | | | | | | |
| **Post-Conditions:** | | A Raft Consensus Config “.rcc” file is created, user is left on dialog asking if they’d like to start one of the nodes | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Open Raft Prototype from the shortcut made on the desktop | | | Raft Prototype program opens on computer |  | | | |
| 2 | Select ‘Create cluster config’ | | | Takes user to Create Cluster Config page |  | | | |
| 3 | Use the example configuration of:   * Cluster name - “My first cluster” * Encryption - Check, set password * Join Retry Attempts - 5 * Persistent Storage - Leave unchecked * Number of nodes - 3 | | | User can change name  User can check Encryption checkbox  User can set a valid password  Retry time auto calculates to 50s |  | | | |
| 4 | Select to ‘Build’ the config, and select where to save the file | | | Config builds successfully  Opens dialog window  File is saved to targeted location |  | | | |

### Beta Test 4 - Create Config - Persistent Storage enabled

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Create Config - Persistent storage enabled | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This is config creation test, it ensures a user is able to build a config describing a config with persistent storage enabled | | | | | | |
| **Pre-Conditions:** | | User has installed Raft Prototype | | | | | | |
| **Post-Conditions:** | | A Raft Consensus Config “.rcc” file is created, user is left on dialog asking if they’d like to start one of the nodes | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Open Raft Prototype from the shortcut made on the desktop | | | Raft Prototype program opens on computer |  | | | |
| 2 | Select ‘Create cluster config’ | | | Takes user to Create Cluster Config page |  | | | |
| 3 | Use the example configuration of:   * Cluster name - “My first cluster” * Encryption - Unchecked * Join Retry Attempts - 5 * Persistent Storage - Set to checked * Number of nodes - 3 | | | User can change name  User can check Persistent Storage  Retry time auto calculates to 50s |  | | | |
| 4 | Select to ‘Build’ the config, and select where to save the file | | | Config builds successfully  Opens dialog window  File is saved to targeted location |  | | | |

### Beta Test 5 - Create Config - Network encryption enabled/Persistent Storage Enabled

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Create Config - Network encryption and persistent storage enabled | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures a user is able to build a config describing a config with network encryption and persistent storage enabled | | | | | | |
| **Pre-Conditions:** | | User has installed Raft Prototype | | | | | | |
| **Post-Conditions:** | | A Raft Consensus Config “.rcc” file is created, user is left on dialog asking if they’d like to start one of the nodes | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Open Raft Prototype from the shortcut made on the desktop | | | Raft Prototype program opens on computer |  | | | |
| 2 | Select ‘Create cluster config’ | | | Takes user to Create Cluster Config page |  | | | |
| 3 | Use the example configuration of:   * Cluster name - “My first cluster” * Encryption - Check, set password * Join Retry Attempts - 5 * Persistent Storage - Set this to checked * Number of nodes - 3 | | | User can change name  User can check Encryption checkbox  User can set a valid password  User can check persistent storage checkbox  Retry time auto calculates to 50s |  | | | |
| 4 | Select to ‘Build’ the config, and select where to save the file | | | Config builds successfully  Opens dialog window  File is saved to targeted location |  | | | |

### Beta Test 6 - Join Cluster - Join from creating config

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Join Cluster - Join from creating config | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This tests that a user can follow on from creating a config to running their first node, to starting more nodes to create a cluster | | | | | | |
| **Pre-Conditions:** | | User has completed one of the Create Cluster use cases and is sitting at a preloaded Start Node screen | | | | | | |
| **Post-Conditions:** | | User starts a node on their computer, user starts other cluster nodes, and is able to join a cluster | | | | | | |
| **Notes:** | | If the user takes too long starting the other nodes, they’ll eventually reprompt to search again | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Select Node1 as the node to start | | | Node1 starts up and searching for the cluster, state is “initialising” |  | | | |
| 2 | Complete “Join Cluster - Join from loading config” Use Case for Node2 | | | Node2 start up and searches for the cluster, finds Node1, creates cluster, one of the nodes starts their UAS |  | | | |
| 3 | Complete “Join Cluster - Join from loading config” Use Case for Node3 | | | Node3 starts up, searches for and joins the cluster. |  | | | |

### 

### Beta Test 7 - Join Cluster - Join from loading config

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Join Cluster - Join from loading config | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures that a user is able to load a node from a config file, and have that node join a cluster | | | | | | |
| **Pre-Conditions:** | | User has completed a Create Config use case and has a Raft Consensus Config “.rcc” file | | | | | | |
| **Post-Conditions:** | | User has been able to start enough nodes to create their described cluster | | | | | | |
| **Notes:** | | If the user takes too long starting the other nodes, they’ll eventually reprompt to search again | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Open Raft Prototype shortcut on the Desktop | | | Program starts up and prompt user for what they’d like to do |  | | | |
| 2 | Select “Start Existing Node”, and select “.rcc” file | | | User is presented with dropdown list of possible nodes to start |  | | | |
| 3 | Select the applicable node to start | | | Node starts up and begins searching for cluster |  | | | |
| 4 | Repeat steps 1-3 for as many more nodes are needed for cluster | | | User starts the rest of the nodes needed |  | | | |
| 5 | Nodes will form and create cluster | | | One of the nodes will show their UAS is started |  | | | |

### 

### Beta Test 8 - Append Entry

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Append Entry | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures that a user is able to attempt to commit a new entry into the distributed log | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type | | | | | | |
| **Post-Conditions:** | | User submits a entry to be committed to the log | | | | | | |
| **Notes:** | | Appending Entries are really requests to append, and are not guaranteed. This is why we aren’t including seeing it committed in this use case. | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Find the Node with their UAS Showing as Running | | | User can identify which node is running the UAS |  | | | |
| 2 | At the bottom of the window type into the Key textbox “Hello”, and the Value textbox “World” | | | User is able to type into the textboxes |  | | | |
| 3 | Select Append | | | The values disappear, a request shoots off in the background to append this entry |  | | | |

### 

### Beta Test 9 - Receive Commit Entries

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Receive Commit Entries | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures that a user is able to observe their entry committed across the cluster | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type and has just pressed Append to append a new entry | | | | | | |
| **Post-Conditions:** | | User is able to observe the entry becoming committed across the cluster | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Look at all the open nodes and observe in their log that “Hello”/”World” has appeared | | | “Hello”/”World” appears in the nodes logs |  | | | |

### 

### Beta Test 10 - Stop Node

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Stop node | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures that a user is able to hard stop a node from participating in the cluster, emulating an unexpected hardware failure of a node to the rest of the cluster | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type | | | | | | |
| **Post-Conditions:** | | User is able to stop a node from participating in the cluster | | | | | | |
| **Notes:** | | This use case only covers the behaviour of the node stopped, observing the surviving cluster will be handled in another use case | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Select “Stop” on one of the nodes in the cluster | | | Node changes to stopped state, perhaps having stopped their UAS to get there depending on which nodes was stopped by the user |  | | | |

### 

### Beta Test 11 - Start Node

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Start node | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This ensures that a user is able to start a node and have it join an existing cluster, emulating recovery from a crashed state | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type, use has stopped the node in question | | | | | | |
| **Post-Conditions:** | | User is able to start a node and have it join an existing cluster | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | Select “Start” on a node in the cluster what was previously Stopped | | | Node starts up, joins cluster and sets it status to “Running” |  | | | |

### 

### Beta Test 12 - Survive Node Failure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Survive Node Failure | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This use case shows that a cluster is able to survive a node failure | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type | | | | | | |
| **Post-Conditions:** | | Cluster survives node failure | | | | | | |
| **Notes:** | | This could also be done on a non-UAS running node, however the nodes do not react to a simple follower failing. Nothing to observe about cluster change, so it’s covered by Stop Node use case. | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | User uses the Stop Node use case to stop the node running the UAS | | | The node stops running it’s UAS  The other remaining nodes are seen to ‘elect’ a new leader among them to run the UAS |  | | | |

### 

### Beta Test 13 - Node rejoin/rebuild distributed log

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Node rejoin/rebuild distributed log | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This use case shows that a node can re enter into a cluster, and bring its log up to date | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type, user has committed some entries to the log, user stops any node | | | | | | |
| **Post-Conditions:** | | Node reneters the cluster and has it’s log brought up to date | | | | | | |
| **Notes:** | |  | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | User uses the Start Node use case to bring back a previously stopped node into the cluster | | | Node finds cluster, the node running the UAS immediately starts populating the log of the previously missing node and brings it up to date |  | | | |

### 

### Beta Test 14 - Read developer logs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | | Read developer logs | | | | | | |
| **Test Type** | | Unsupervised User Acceptance Test | | | | | | |
| **Test Description** | | This use case to show off the levels of logging the raft algorithm is doing in the background | | | | | | |
| **Pre-Conditions:** | | User has created a cluster of any type, user has committed some entries to the log | | | | | | |
| **Post-Conditions:** | | User is able to observe developer logs | | | | | | |
| **Notes:** | | This information can be useful for debugging any issues that may occur. The debug level is set when using the code, and defaults to debugging at info level | | | | | | |
| **Results** | |  | | | | | | |
| **Step.** | **Step Description** | | | **Expected Result** | **Result** | | | |
| 1 | User picks any nodes and selects to move over to the Log tab | | | User is able to view and scroll through the back end developer log of the consensus algorithm |  | | | |
| 2 | Use selects Debug or Trace from the dropdown and sees the flow of messages of flying by that the algorithm is writing to the log for debugging purposes | | | User is able to view and scroll through the back end developer log of the consensus algorithm, they can now see heartbeat and message flow/processing |  | | | |

### 