**Master Test Plan**Team Decided - Raft Consensus Library

## Testing Strategy

### Test Driven Development (TDD)

Test driven development will be utilised in this project; this methodology focuses on testing the design specifications is accurately implemented in the code, rather than testing just the implementation of code itself. If all of these test are written and pass, then it can reliability be confirmed that the code is accurate to the specification, no more no less.

### Unit testing (UT)

These tests are directly related to the use of TDD, and confirm that code is meeting the requirements of each unit’s functionality. Unit testing is the set of atomic tests on the implementation of each object’s public contract to ensure they meet requirements. If all of these tests are written and pass, then it can be reliably be confirmed that each unit of code is correctly performing it’s functionality.

### Integration Testing (IT)

These tests are directly related to the use of TDD, and confirm that code is meeting the requirements of the design’s use cases. These IT are simply groups of UTs which form together to make a use case. If all of these tests are written and pass, then it can be reliably be confirmed that the code is performing all the use cases to specification.

### Code review (CR)

During development, strict adherence to coding guidelines greatly improve maintainability with a flow on effect of code reliability. To perform code review, a developer writes or makes changes to the code base, and then a separate developer reviews and audits line-by-line those changes, ensuring quality and implementation ideas match design. This reduces, and ideally removes, obvious logic errors, code smells, improper implementation and other various code issues.

### Prototype (PT)

The prototype will include a minimal functional feature set of the library which can be used to confirm successful implementation of the consensus algorithm. This prototype will also be confirmed by the TDD life cycle.

### Implementation Testing (IMPT)

Alpha/beta testing is not within a reasonable scope of our project due to the unreasonable level of burden it would place on a alpha/beta tester to integrate our library. To solve this the library will be integrated by ourselves into an open source program to confirm all integration functionality.

### Design validation (DV)

Some functionality is not so black/white where it can be confirmed through software tests, for this we have the professional work of developers confirming functionality. For example, minimal resource usage cannot be unit tested, however a developer can give their professional results confirming the matter.

## Tests to be Conducted

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Feature/functionality** | **Test method** | **Testing environment** | **Acceptance criteria** | **Role** | **Planned Stage** |
| 1 | Can write to debug log | UT | VS | UT pass | Joshua | Prototype |
| 2 | Can write to distributed log | UT | VS | UT pass | Sean | Prototype |
| 3 | Can read from distributed log | UT | VS | UT pass | Joshua | Prototype |
| 4 | Can send and receive messages from other nodes | UT | VS | UT pass | Sean | Prototype |
| 5 | Can communicate using encrypted messages | UT | VS | UT pass | Joshua | Prototype |
| 6 | Node authenticates using zero knowledge password proof | UT | VS | UT pass | Sean | Prototype |
| 7 | Does library call UAS start/stop | UT | VS | UT pass | Joshua | Prototype |
| 8 | Can hold successful election | UT | VS | UT pass | Sean | Prototype |
| 9 | Bring node log up to date | UT | VS | UT pass | Joshua | Prototype |
| 10 | Falls to follower when detecting newer leader | UT | VS | UT pass | Sean | Prototype |
| 11 | Can maintain service during node failure/loss | UT/IT | VS | UT pass, IT pass | Joshua | Prototype |
| 12 | Can recover from node failure/loss | UT/IT | VS | UT pass, IT pass | Sean | Prototype |
| 13 | Consensus between distributed systems | IT | VS | IT pass | Joshua | Prototype |
| 14 | Fault tolerant distributed service | IT | VS | IT pass | Sean | Prototype |
| 15 | Security | IT | VS | IT pass | Joshua | Prototype |
| 16 | Privacy | IT | VS | IT pass | Sean | Prototype |
| 17 | Cross Platform | DV | Developer assessment | Two developers confirm functionality | Joshua | Prototype |
| 18 | Mitigate project abandonment | DV | Developer assessment | Two developers confirm design | Sean | Prototype |
| 19 | Compatibility | DV | Developer assessment | All code is only entered into  repository upon successful review from code review | Joshua | Prototype |
| 20 | Troubleshooting | DV | Developer assessment | Two developers confirm functionality | Sean | Prototype |
| 21 | Improved reliability of existing service | IMPT | VS/Workstation manual test | Reliability of existing service is  improved, confirming still responds during node failure | Joshua | Prototype |
| 22 | Complete proven reliability | IMPT | VS/Workstation manual test | Confirm function after all node  scenario failures | Sean | Prototype |
| 23 | Minimal overhead/impact to service performance | IMPT | VS/Workstation manual test | Two developers confirm functionality | Joshua | Prototype |
| 24 | Usability | IMPT | VS during implementation | Two developers confirm functionality | Sean | Prototype |
| 25 | Availability | IMPT | VS/Workstation manual test | Two developers confirm functionality | Joshua | Prototype |
| 26 | Confirm library is one-click integratable from Nuget | IMPT | VS during implementation | Two developers confirm result | Sean | Prototype |
| 27 | Can do CRUD operations on Nodelist | UT | VS | UT pass | Joshua | Version 1.0 |
| 28 | Can propagate node info | UT | VS | UT pass | Sean | Version 1.0 |
| 29 | Ability to attempt to designate a node to run the UAS | IT | VS | IT pass | Joshua | Version 1.0 |
| 30 | Scalability | IT | VS | IT pass | Sean | Version 1.0 |
| 31 | Minimal additional surface area for failure | CR | Bitbucket Pull Request | All code is only entered into repository upon successful review from code review | Joshua | Version 1.0 |
| 32 | Minimal resource usage | DV | Developer assessment | Two developers confirm results | Sean | Version 1.0 |
| 33 | Performance | DV | Developer assessment | Two developers confirm results | Joshua | Version 1.0 |
| 34 | Upgrade path | IT | VS | IT pass | Sean | Final |
| 35 | Quality | CR | Bitbucket Pull Request, Pair Programing, Resharper | Two developers confirm functionality | Joshua | Final |
| 36 | Documentation | DV | During implementation testing | Two developers confirm documentation  existence | Sean | Final |
| 37 | Testability | DV | During implementation testing | Two developers confirm functionality | Joshua | Final |
| 38 | Extendability | DV | Developer assessment | Two developers confirm design | Sean | Final |
| 39 | Auditability | DV | Developer assessment | Two developers confirm functionality | Joshua | Final |
| 40 | Reliability | IMPT | VS/Workstation manual test | Two developers confirm results | Sean | Final |
| 41 | Practical implementation full functionality working as required | IMPT | Developer assessment | Two developers confirm results | Joshua | Final |