Changelog				
Project: Team Decided Raft Consensus				
Please note, hyperlinks in Section are linked to nearest subheading				
ID	**Section**	**Type**	**Description**	**Notes/Justification**
	Project Vision, User Environment, Example Scenario 1	Clarity	Specifying .NET core server to drive home target market	Clarity
	Project Vision, User Environment, Example Scenario 1	Clarity	Directly specifying how example scenario 1 would make use of this library	Clarity
	Project Vision, User Environment, Example Scenario 2	Clarity	Rephrase sentence for clarity	Clarity
	Project Vision, Product Overview, Needs and Features (Functional requirements)	Clarity	Added hyperlinks to Prototypes to our Nuget package	Clarity
	Project Vision, Product Overview, Needs and Features (Functional requirements)	Project schedule update	Moved Log Compaction requirement to Version 1.0 instead of Prototype	We think this is still an important feature, however it does not fall on the critical use case which is the requirement of Prototype milestone. This will be implemented later.

	Project Vision, Product Overview, Needs and Features (Functional requirements)	Scope management	Dropping Warm Node feature	We don't believe this is an important feature to the product, perhaps it may be included later in an extended life cycle of the library, however we won't be adding it for this project. The use case just isn't that critical.
	Project Vision, Product Overview, Needs and Features (Functional requirements)	Scope management	Changing Ability to Pick Ideal Leader to "Ability to attempt to designate a node to run the UAS"	Picking an ideal leader is very hard to implement, and heavily dependant on how each UAS defines fitness. We think it's better to leave that scope to the UAS developers, however we then need to add the ability to attempt to make a node the leader of the cluster. They can use this on their fittest server to start/takeback UAS.
	Project Vision, Product Overview, Needs and Features (Non-functional requirements)	Clarity	Added hyperlinks to Prototypes to our Nuget package	Clarity
	Project Vision, Product Overview, Needs and Features (Non-functional requirements)	Project schedule update	Changing the release of the Nuget package to Prototype from Final	Part of the requirements for the prototype were "deployed in the intended production environment", this change directly ticks that requirement
	Project Vision, Product Overview, Needs and Features (Non-functional requirements)	Scope management	Removing Network Agnostic option from compatibility non-functional requirement	We don't believe this is an important feature to the product, perhaps it may be included later in an extended life cycle of the library, however we won't be adding it for this project. The use case just isn't that critical. Although it's a relatively simple add, to confidently consider it working we'd need to implement and tests a second networking stack using it.
	Project Vision, Product Overview, Justification	Clarity	Adding the caveat to using the library in a 30-60 tick game of needing performance analysis to be run first	Although, calculations of round trip time conditions can show that this is possible, we're aware the algorithm is currently running too slowly and needs performance analysis to be

				able to function at this level.
	Project Vision, Product Overview, Justification	Clarity	Rewriting to paragraph about justifying removing Ability To Pick Ideal Leader feature	As stated above, implementing the feature is unreasonably cumbersome and UAS specific. So we'll be leaving this up to, and enabling the ability for, the people to implement the ideal leader selection themselves.
	Project Vision, Product Overview, Justification	Scope management	Deleted paragraph justifying Warm Nodes	As stated above, we don't believe this feature is so important
	Requirement model, Use Case Model	Update	Remove "Send Message" and "Receive Message" use cases	This was part of the network agnostic feature we were going to implement, however as stated above we've removed this feature as it's been deemed not important enough to implement
	Requirement model, Use Case Model	Update	Remove "Request Fitness" and "Respond to Fitness" use cases	This was removed as we've changed from implementing the design of fitness calculation ourselves, and offloaded that ability to the UAS developer through enabling a "Attempt to Run UAS" use case for them to use when doing their own fitness algorithms if they'd like.
	Requirement model, Use Case Model	Update	Added "Attempt To Run UAS" use case	We added this use case to allow developers to implement their own fitness algorithms to decide which node should run the UAS
	Requirement model, Use Case Model	Update	Remove "Run Cluster" use case	We were previously viewing this from a perspective that a node needs to make a cluster for it to exist for nodes to join. We've now evolved our view to now being that each node simply tries to join the cluster it's been instructed to, simply assuming it exists. And when many nodes talk to each other, and they're all assuming it exists, they start the cluster. This means developers only need to use the one "Join cluster" use case now

	Requirement model, Use Case Model	Update	Remove “Confirm Identity” use case	Our initial plan was to have identities based on certificates, meaning the security on the communication was based on the UAS Confirming the identity of any joining nodes. However, the implementation was more difficult than it was worth, so we’ve now changed to the security being based on zero knowledge password proofs (which is using hashes with nonce’s to conduct a two way challenge/response)
	Requirement model, Use Case Model	Update	Changed “Start Server” and “Stop Server” use case names to “Start UAS” and “Stop UAS”	Change terminology to better describe the action required and implemented.
	Requirement model, Use Case Model	Update	Changed “Read Entry Value” use case to “Read Log”	Change terminology to better describe the action required and implemented
	Requirement model, Short Use Case Descriptions	Update	Remove “Send Message” and “Receive Message” use cases	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Short Use Case Descriptions	Update	Remove “Request Fitness” and “Respond to Fitness” use cases	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Short Use Case Descriptions	Update	Remove ”Run Cluster” use case	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Short Use Case Descriptions	Update	Added “Attempt To Run UAS” use case	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Short Use Case Descriptions	Update	Remove “Confirm Identity” use case	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model

	Requirement model, Short Use Case Descriptions	Update	Changed “Start Server” and “Stop Server” use case names to “Start UAS” and “Stop UAS”	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Short Use Case Descriptions	Update	Changed “Read Entry Value” use case to “Read Log”	Additional changes required in Short Use Case Descriptions from changes above made to Use Case Model
	Requirement model, Domain model	Update	Updated Domain model to include the current implemented consensus API	Changes were required as we’d updated the underlying API, so this needed to be reflected in the Domain Model
	Requirement model, Discussion regarding CCRD	Reflection	Added reflection paragraph on how we have cover CCRD	Clarity
	Final Architecture, Decisions, Constraints, and Justifications,Universally standard data structure for distributed log	Clarity	Re-wote discussion on Key Value structure used in our implementation.	Expressed the log better as a key value data structure, and explained the choice against using Raft’s normal List style log.
	Final Architecture, Decisions, Constraints, and Justifications,Task/callback style API	Clarity	Added section discussing the asynchronous design of methods	Clarity primarily, but we noticed it as an important design consideration which had been omitted
	Final Architecture, Architectural Mechanisms, Architectural Mechanism 4 - Security	Update	Qualify the way in which we implement security within the library.	We are no longer using manual verification and certificate based authentication, we’re moving onto a simpler to implement, and simpler for usability reasons of password based authentication.

	Final Architecture, Layers or Architectural Framework	Update	Update image to reflect API implementation	Changes were required as we'd updated the underlying API, so this needed to be reflected in the diagram
	Risk List Generic Project Risk	Update	Changed Scope creep inflates scope probability to "Medium" from "Low"	Although during our current work we've found that we're reducing scope due to time restrictions, rather then scope creeping, we're aware that next session is all about adding features. During our first session we became aware during implementing features that we had to add additional supportive features as well, showing that features can quickly get away from you, hence our change to "Medium".
	Risk List Generic Project Risk	Update	Estimates for milestones are inaccurate, marking probability to "High" from "Medium"	As we can see from our iteration plans the only consistent thing we could predict is that we'd predict incorrectly. We did however innact the Contingency Plan of spending time trying to iteratively identify root causes, and better calculations for estimating. We improved our estimations from 400% out to 110% out, and we can more reliably predict times using the "double it, plus a bit" method. This root issue is we don't have the real world experience to accurately estimate how longs things will take yet.
	Risk List Generic Project Risk	Update	Changing "Finish project too early" risk to "Low" probability from "Medium"	We've simply got too many nice to have features we'd like to add and not enough time to add them all, so it's now about picking the desired features. We won't finish early.
	Generic Project Risk	Update	Closed the risk of "Technology components have security vulnerabilities"	This is because we're now using Microsoft's own implementation of security algorithms, these are patched automatically through Windows Update.
	Generic Project Risk	Update	Reduced risk of "Code quality issues" from "High" to "Medium"	Using Style Guides, Peer Programming, completed a bunch of exhaustive tests through development find bugs, and also have unit testing/integration testing on major points

	Generic Project Risk	Update	Closed risk of “User acceptance failure”	We’ve successfully implemented this library into a standalone Prototype which was built with only the Nuget package, we also made and showed it working properly with an installer.
	Project Specific Risk	Update	Changing “Poor software quality” probability from “High” to “Medium”	Using Style Guides, Peer Programming, completed a bunch of exhaustive tests through development find bugs, and also have unit testing/integration testing on major points
	Project Specific Risk	Update	Closed risk of “Security too complex”	We’ve successfully implemented security into the library. It was initially too hard, as we predicted, so we changed to using a zero knowledge proof authentication method, and Microsoft’s own security algorithm implementations rather than a library
	Project Specific Risk	Update	Closed risk of “Prototype failure”	We’ve successfully implemented this library into a standalone Prototype. We recognised that it’s probability of occurrence was medium, with a high impact, so we mitigated this through starting this as early as possible.
	Project Specific Risk	Update	Changing “Multithreading introduces high level of difficult in troubleshooting” probability and risk from “High” to “Extra High”	The level of difficulty involved in troubleshooting multithreaded errors/bugs etc, is far beyond any reasonable expectation placed upon a student project. The effort and difficult of this has forced us to research and design mechanisms and tools to aid in the process (i.e. intrinsic logging, and learning visual studio multithreaded step through processing). There is no way we could have predicted simply how difficult this issue was for us, and the time sink it was during the debugging process. When we initially set up with 3 threads per node for networking, 2 threads per node for consensus and the main thread, we had times troubleshooting errors where there was up to 50 running threads. We had to heavily refactor focusing on code quality, deadlock avoidance techniques and reduction of threads.

	Project Specific Risk	Update	Closed risk of “Multithreading introduces high level of difficult in troubleshooting”	Although it’s been absurdly difficult, there are currently no bugs or deadlock issues in the code we’re aware of after extensive testing. We aren’t looking to implement any more features which change the functionality of any multithreaded flows, so we’re confident we won’t have issues in this area again.
	Project Plan	Update	Set E-4 iteration final date to 20/06	Jim graciously gave us an extension of two weeks for us to complete our work within, this is reflective of that. As E-4 was targeted for contingency of the E-3 iteration, the accuracy of this subject’s Project Plan still stands.
	Project Plan	Update	Changing our **C-1** iteration next session to be focusing on code quality, rather than implementing the Dynamic Cluster Membership feature straight away	Keeping our risk list in mind, we can see the biggest risks to our project currently primarily revolve around code quality and code complexity. As we’ve learnt far more in this session regarding unit testing, we’d like to implement a redesigned unit testing framework which allows us to easily test all different node member variations (2, 3, 5, 7, 9, etc. nodes). As our primary CCRD has already been completed, and we’re into the “feature adding” stage now, it’s reasonable for us to take an iteration for further testing as there is no more “must have” features of the software.
	Project Plan	Update	Changing our **C-2** iteration next session to be the original C-1, which was implementing Dynamic Cluster Membership	This is a feature we’d really like to have, so we’d like to make sure it gets completed. We’ve currently got a code base at the moment which passes testing, so any changes to it should be verified to not break existing functionality, this is why we aren’t trying to introduce the new feature until after our new C-1 iteration of redesigned testing.
	Project Plan	Update	Changing our **C-3** iteration next session from “Upgrade Path and Performance Analysis” to “Log Compaction”	This is a feature which we think a reasonable user of the software can expect. That after using the distributed log for a long time it does not run into file size limits. We’d like to implement this feature straight after Dynamic Cluster Membership.

	Project Plan	Update	Changing our **C-4** iteration next session. We've added optional extra nice-to-have use cases after contingency allowance and IOCM work	We've identified that we'd also like to complete these extra nice to have features as part of our requirements, even though they're low priorities. We've added them to the end of the iteration for in case we have time, we're hopeful we will have the time. We've maintained the C-4 iteration as contingency for previous iterations as well.
	Project Plan	Update	Changing our **T-1** iteration, removing the implementation of our project into an open source project, and instead we'll more simply integrate our new features into our existing prototype application	Developing the prototype took a huge amount of effort and time, and that was for code that we had full control over and understanding of. We believe that successful reliable integration into an open source project would be exhaustive time consuming and far beyond any reasonable expectation for this student project. However we'd still like to show off our new features added this session, so we'll simply integrate them into our existing Prototype and repurpose that program as a Demo Application of the library. We'll publish the Demo Application open source as well.
	Project Plan	Update	Changing our **T-2** iteration. Adding in finishing off the prototype integration, adding producing user documentation for library	We're adding some contingency for completion of integrating our new features into the prototype. We're also adding in that we should produce our public user documentation for our library (including XML documentation for publicly accessible methods) and example code in the readme.md version control file
	Master Test Plan	Major rewrite	Full re think of testing methods, environment and acceptance criteria	- Changed Design Validation to Developer validation - Removed unnecessary and duplicate tests, and tests which were for features we've removed - Added the Planned Stage column for when these tests are due to be performed - Rewrote the names of tests to better clarify - Ensured that tests link closely to use cases, functional and non-functional requirements - Added Demo test type, so we can use a demo program to show functionality