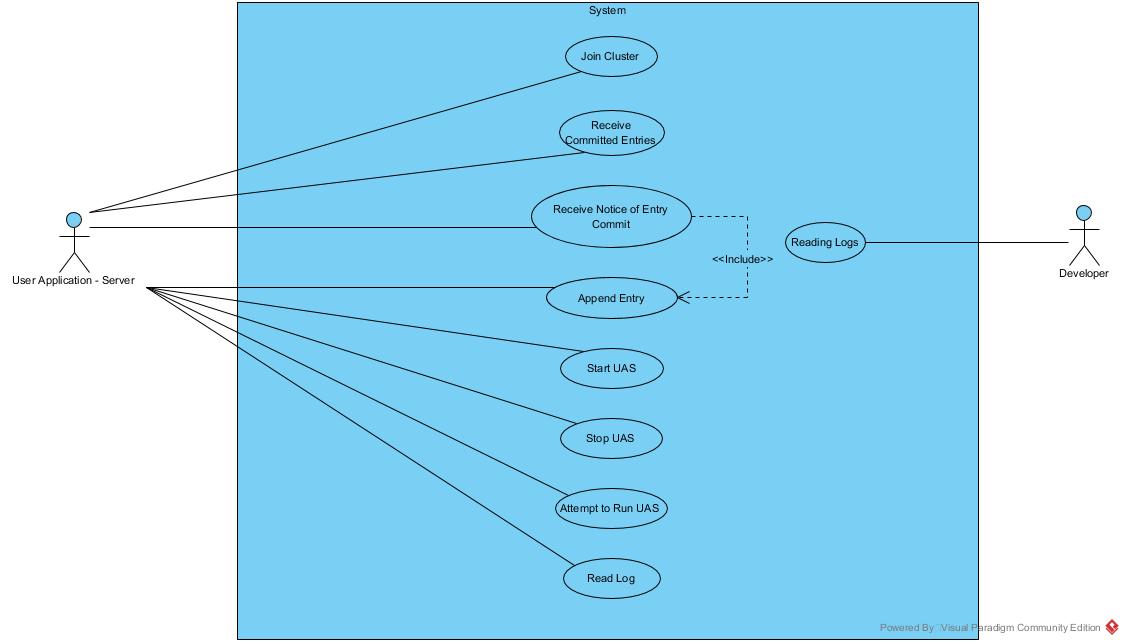
### Requirement Model

### Use Case Model

After investigation of the requirements the following [Use Case diagram](https://media.discordapp.net/attachments/418342556800385048/433932325508743168/2_-_Use_Case_Diagram.png?width=720&height=585) has been developed to outlining the initial set of interactions required

.

### Short Use Case Descriptions - In Table Form

#### Actor Key

UAS = User Application Server

Dev = Developer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case Name** | **Precondition** | **Actor** | **Need** | **Do**  **Something** | **Basic Intent/Goal** |
| Join Cluster | Has information about the cluster | UAS | Join the cluster | Join Cluster | Become a member of the cluster |
| Receive Committed Entries | The current node has joined the cluster | UAS | Bring itself up to date | Receive Committed Entries | Can maintain consensus |
| Receive Notice of Entry Commit | The current node has attempted to append an entry | UAS | To know if the message they requested to be committed, was committed | Receive Notice of Entry Commit | So that they may update the running UAS state |
| Append Entry | The current node should be running a UAS | UAS | To add a message into the distributed log and have it protected by consensus | Append Entry | Message gets committed into the consensus |
| Start UAS | The current node was previously not running the UAS | UAS | To be notified when they need to start running the UAS | Start UAS | They may go ahead and start up their UAS to provide services |
| Stop UAS | The current node was previously running the UAS | UAS | To be notified when they need to stop running the UAS | Stop UAS | They stop acting as a UAS and providing service |
| Attempt to Run UAS | The current node currently doesn’t run UAS | UAS | Attempt to designate a specific node to run the UAS | Attempt to Run UAS | The identified nodes will try to become leader and hence be responsible for running the UAS |
| Read Log | The current node has joined the cluster | UAS | Read log entries that have been committed | Read Entry Value | Can reference the committed data |
| Read Logs | At least 1 node has been instantiated | Dev | Understand what the underlying consensus algorithm is doing, perhaps tracking a bug | Read Logs | Understand what the underlying consensus algorithm is doing |

### 

### Domain model

#### 

#### Discussion regarding domain model

We’ve found this is easiest to explain using an example of a distributed multiplayer game running a which is in a server/client setup. In this example the server is also running on one of the clients. In that example, the UAC would be the game client which the player’s would be interacting with, and UAS would be the game server all the game clients are talking back to. Our library would be integrated into the UAS.

Reading from the top left of the diagram it’s shown how the UAC (i.e. game clients) talk to the active UAS (i.e. game server) in the Cluster Application Servers(CAS). Then it’s shown how the CAS is made up of UASs. It’s the library which links these UACs together into a cluster, and the UACs simply talk to whomever is the ‘leader’ UAS in the CAS. The UAC can simple just try each of the IPs in the CAS until they find the leader who responds, this cuts down on the complexity in the UAC software.

Next, the UAS uses the Consensus API to talk to it’s internal Consensus Node. This consensus node is the part that handles cluster operations, and part of that is instructing the UAS to let it know when it should and shouldn’t be running the active game server. The nodes all communicate to each other through messages which are sent and encrypted by the Networking Library. Consensus between the nodes is through a shared distributed log (DistributedLog) which the active UAS commits entries into, and the follower nodes are updated with. Each node stores information about the other nodes which are in the cluster with it. And each node does verbose logging for troubleshooting reasons.

#### Does the analysis of non-functional requirements demonstrate an ability to identify, quantify, prioritise, and communicate required system qualities? (LO2)

This is the part of the document where the above would be analysed, however after discussing with Jim, he has confirmed that this is covered by our ”Business Justification for Functional and Nonfunctional Requirements” section in the Project Vision above.

#### Discussion regarding CCRD

The CCRD Use Case is the list of the most important (critical, core, risky, difficulty) Functional/Non-functional requirements, and as per our project milestones, must be implemented into the Prototype. As our project fundamentally is very atomic (i.e. it either works or it doesn’t), we’ve had to include all but a single Use Case (Attempt To Run UAS) into our Prototype to mitigate the risk of implementation issues later on. This means our CCRD Use Case includes 7 of the 11 requirements of the Prototype (As outlined in Project Vision), and as stated above includes all but 1 of our above outlined use cases.