Feature checklist

# Assessment Rubric

### Submission requirements (all non non-submissions)

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Submission requirements** |  |
| R1 | Viva attended | Will Do (viva is after submission) |
| R2 | Checklist provided to tutor at viva | Will Do (viva is after submission) |
| R3 | Submission is a single zip file | Done |

### Basic Requirements Required for Pass (40% or greater)

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Submission requirements** |  |
| R4 | Submission contains appropriate folders for  each server or client type created | Done |
| R5 | Code runs on any computer in MS214 or MS215.  **Note:** Only the address of the bootstrapping/DNS node for your system may be hardcoded, no other paths or IP addresss may be hardcoded. Any path should be relative to the location of the executable | Done - The code does run on any computer in MS214 or MS215. With the bootstrap node IP being the only hardcoded address. |
| R6 | Solution interprets/builds without any errors  or warnings | Done – solution builds and runs without any errors or warnings. |
| R7 | Implementation Log provided in zip file | Done – in the submission folder to read |
| R8 | Testing and analysis log provided | Done – in the submission folder to read |

|  |  |  |
| --- | --- | --- |
| R9 | Feature checklist provided in zip file | Done – in the submission folder to read |
|  | **System/Distribution Requirements** |  |
| S1 | Bootstrap node is provided that allows other nodes on the network to register their name,  IP address, and port. | Done - Each node has its unique address and name. All content nodes and functional nodes connect and register with the bootstrap node. |
| S2 | Bootstrap node is provided that allows other nodes on the network to anonymously retrieve IP address and port by providing a  name | Done - Each nodes in the network can anonymously retrieve IP address and port by requesting. |
| S3 | Connection and disconnection are handled  without errors on the bootstrap node. | Done – all connections and disconnections are handled without any error. |
|  | **Functional Requirements** |  |
| F1 | Function node is provided that can register  with bootstrap node | Done - authentication node and microservice handles registering |
| F2 | Function node is provided that can  authenticate the connecting client | Done - authentication node and microservice allow clients to authenticate their login using authentication token system |
| F3 | Function node is provided that can provide a  list of audio files to a connected client | Done - Functional node is named a file distribution node and microservice can send the full list of all available music/audio |
| F4 | Function node is provided that can transfer  an audio file to the connected client | Done - Functional node is named file distribution node and microservice can send the music/audio requested by the client. |
|  | **Client Requirements** |  |
| C1 | Client can connect to bootstrap node to get  address of functionality nodes | Done – Client connects to bootstrap and it gives the address of the functionality nodes |
| C2 | Client can authenticate and/or register with  function node to create a persistent account | Done – client can log in or register |
| C3 | Client can retrieve and display a list of audio  files on the function node | Done – once logged in the client can get a list of all audio/music files |
| C4 | Client can download an audio file from the  function node | Done – client can select a specific audio/music file and download it |
| C5 | Client can playback uncorrupted audio file  without errors | Done - The MD5 checksum is utilised to verify the integrity of the download with the FDN microservice calculating and dispatching both the checksum and the file to the client, who then recalculates the checksum post-download; a match between the two checksums confirms the success of the file download. |

#### Additional Features Required for Grade of 50% or Greater

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **System/Distribution Requirements** |  |
| S4 | Non-dynamic load balancing is implemented | Done – Non dynamic load balancer implemented |
| S5 | Nodes can be spawned and managed dynamically (non-manual replication) on a  single machine | Done - nodes can be replicated on a single machine. |
|  | **Functionality Requirements** |  |
| F5 | Functionality nodes provide separation of services e.g. authentication node, centralised  load balancer, content node | Done - Bootstrap node provides control node with requests on what nodes they should spawn into depending on what the system needs either as a functional node of authentication node or file distribution nodes and these functional nodes spawn microservices. |

#### Additional Features Required for Grade of 60% or Greater

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **System/Distribution Requirements** |  |
| S6 | Dynamic load balancing is implemented | Not implemented |
| S7 | Nodes can be spawned and managed  dynamically (non-manual replication) | Done - Spawned by control nodes, functional nodes authentication and file distribution and microservices |
|  | **Functionality Requirements** |  |
| F5 | Management node system is implemented to allow dynamic spawning of functionality  nodes on remote machines | Done - Bootstrap handles the management informing control nodes on what it is required to turn into then the control nodes dynamically spawn functionality nodes and these nodes spawn microservices. |

#### Additional Features Required for Grade of 70% or Greater

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **System/Distribution Requirements** |  |
| S8 | Functionality nodes are replaced with  “microservices” – a load balanced collection  of nodes that shares data/function. | Done – Microservices is implemented functional nodes (auth, fdn) spawn Microservices which handle login/signup/token validation, music/audio list and audio/music download. |
|  | **Client Requirements** |  |
| C6 | Client can validate that the file is not corrupted through the use of error checking  such as an MD5 checksum. | Done - File distribution handles this the microservice calculates a checksum and transmits it to the client with the audio file. The client calculates its own checksum from the file download and compares it to the supplied checksum. |

#### Additional Features Required for Grade of 80% or Greater

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **System/Distribution Requirements** |  |
| S9 | Single additional feature e.g. peer-to-peer caching, consistent hashing, or additional platform clients e.g. web or mobile client, or  other functionality agreed with your tutor | Implementation of a GUI front end using tkinter the client/user can use to navigate and use the program through. |

#### Additional Features Required for Grade of 90% or Greater

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **System/Distribution Requirements** |  |
|  | Multiple additional features e.g. peer-to-peer caching, consistent hashing, or additional platform clients e.g. web or mobile client, or  other functionality agreed with your tutor |  |

|  |  |
| --- | --- |
| **List any other features implemented** | **Details** |
| User Authentication token | Once a client has logged/signed in once and tries to login again, token has been saved by bootstrap this allows a quicker login for the client. |