**Features Checklist**

**Student Number: 100723654**

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

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Submission requirements** |  |
| R1 | Viva attended | Provided video |
| R2 | Checklist provided to tutor at viva | Provided in the zip |
| R3 | Submission is a single zip file | Yes |

1. **Basic Requirements**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Submission requirements** |  |
| R4 | Submission contains ‘Client’ and ‘Server’ folders inside the zip file | Yes – Submission contains ‘Server’ and ‘Client’ folders. |
| R5 | Code runs on any computer in MS214 or MS215.  **Note:** you should not hardcode a path or IP address in the code that you write. Any path should be relative to the location of the executable | Yes – Code runs locally on address ‘127.0.0.1’ and port 8080 |
| R6 | Solution interprets/builds without any errors or warnings | Yes – The application interprets/builds without any errors or warnings. |
| R7 | Implementation Log provided in zip file | Yes – Provided Implementation log docx, readme.txt, requirements.txt, testing docx, features checklist docx and video link |
| R8 | Feature checklist provided in zip file | Yes- Featurelist is provided. |
|  | **Networking Requirements** |  |
| N1 | Synchronous bidirectional communication between client and server. | Yes - Implemented bidirectional communication using sockets -> Server.py listen function and Serverlib.py read/write functions |
| N2 | Connection and disconnection are handled without errors on the server side. | Yes - Connection and disconnection handling without errors-> Server.py close\_client function and Serverlib.py inside read function |
| N3 | Error handling and message content verification are handled on the server side. | Yes - Error handling and message verification -> ServerLib.py read/write functions |
|  | **Security Requirements** |  |
| S1 | Network traffic is encrypted using a standard algorithm using [a pre-shared, entered, or non-negotiated key]. | Yes - Network traffic encrypted using AES-GCM with shared key exchange at start (Server.py listen function, Client.py connect\_to\_server function) |
|  | Application Requirements |  |
| A1 | Implements at least 2 states that alter the behaviour of the system based on user input. | Multiple Client states (login, signup, dashboard, create/edit task) UI updates (ClientStateMachine.py, GUI.py) |

1. **Additional Features**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Networking Requirements** |  |
| N4 | Multiple clients supported by a single server. | Yes - Server supports multiple simultaneous connections via thread creation for each client |
| N5 | Error handling and message content verification are handled on both the client and server side (replaces N3) | Yes - Extensive error handling on both server and client across all files |
|  | **Security Requirements** |  |
| S2 | Key is negotiated using an appropriate mechanism such Diffie-Hellman (modifies S1 component in []) | Yes - Key exchange protocols implemented: RSA public key exchange, then AES key sent encrypted to client |
|  | Application Requirements |  |
| A2 | Implements at least 1 complex state that allows data to be stored between sessions. | Yes - Multiple Client states (login, signup, dashboard, create/edit task), SQLite persistent storage, instant task updates, persistent user accounts/roles |

1. **Additional Features**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Networking Requirements** |  |
| N6 | Asynchronous bidirectional communication between client(s) and server (replaces N1). | Yes - Threaded implementation in Serverlib.py and ClientLib.py (readThread/writeThread creation) |
|  | **Security Requirements** |  |
| S3 | Packets implement mechanisms to prevent at least 1 kind of attack e.g. replay attacks. | Yes - Protection against replay and Man in The Middle attacks via RSA/AES implementation with aesIV and authtag verification |
|  | Application Requirements |  |
| A3 | Implements at least 1 complex state that alters typical network communication pattern (e.g. client can send an arbitrary number of messages in a row) . | Yes - Multiple clients can update tasks, notification system broadcasts (UI popup limitation on same computer), CRUD operations in any order, dashboard allows multiple operations |

1. **Additional Features**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Networking Requirements** |  |
| N7 | Shared state information between multiple clients and server. | Yes – Tasking sharing between clients, notification system, admin view users. |
| N8 | Network code is provided as a layered solution separated from the application logic. | Yes, Layered network implementation |
|  | **Security Requirements** |  |
| S4 | AAA implemented (authentication, authorisation, and auditing) e.g. user privilege levels. | Yes - Admin and user roles implemented with MD5 password hashing and required login (StateMachine.py, Authentication.py, Database.py, ServerLogger.py) |
|  | Application Requirements |  |
| A4 | Implements an appropriate way to view or access AAA information. | Yes - Admin user/role viewing, IT department access to server\_log.log and client\_log.log with comprehensive logging |

1. **Additional Features**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Networking Requirements** |  |
| N9 | Single additional feature e.g. heartbeat, peer-to-peer implementation, or other functionality agreed with tutor. | No. |
|  | **Security Requirements** |  |
| S5 | Single additional feature e.g. data encrypted on storage, user password resets, account lockouts. | Partially - Passwords stored with MD5 hashing |

1. **Additional Features**

|  |  |  |
| --- | --- | --- |
|  | **Requirement** | **Level of Implementation/Details** |
|  | **Networking Requirements** |  |
| N10 | Multiple additional feature e.g. heartbeat, peer-to-peer implementation, or other functionality agreed with your tutor (replaces N8). | Yes - Asynchronous bidirectional communication, notification system, graceful connections/disconnections, buffer message system, packet header verification |
|  | **Security Requirements** |  |
| S6 | Multiple additional feature e.g. data encrypted on storage, user password resets, account lockouts (replaces S5). | No. |

|  |  |
| --- | --- |
| **List any other features implemented** | **Details** |
| Graceful server shutdown | The server closes gracefully. |
| Good logging system |  |
| Good GUI | Invested a lot of time of the visual aspect of the application for login/signup and especially the dashboard. |
| Dynamic task display | Tasks are showed dynamically. Each task creates a frame that gets placed below as a stack |
| Dynamic Button creation | Tasks create edit and delete buttons for each task that associates functionality to the specific task. |
| Implemented support for concurrent operations with thead-safe locking. | When threads have to use resources such as queues and the database, they accumulate a lock and release it when finishing their job. |