# Protocol Design

## Session Establishment

1. Client will first make a request to establish a connection to the server. The server will then accept, thus establishing a connection.
2. Client will send a server the username of the account it wants to login into
3. Server will response with a token, a randomized string of characters
4. Client will then append their password to the token, generating a md5 hash, sending this has to the server.
5. The server will generate its own respective hash with the password it has stored.
6. If the hashes are identical, the client will the be authenticated, otherwise the client is disconnected.

## Error Handling, Timeout

1. All connections, sending, receiving packets will be coded inside of try-catch statements, allowing us to debug our code easier, and create a user-friendly experience. This would also make it easier to prevent both the client and server program from crashing during unexpected test cases.
2. A completed packet is defined as a packet, with the correct header information and ending sequence, denoted as “\n\n”. If a packet has an unrecognizable header or an incorrect ending sequence, then it is not considered a valid packet
3. When sending and receiving packets, if either client or server isn’t able send and acknowledge valid packets within 5 seconds sending and receiving, then we will assume that the packet is lost and needs to be recent. There will be print statements notifying the user if their packages have been received or lost.

## Concurrency Handling

1. To handle multiple client transactions, we will be running a separate thread per client.
2. Once a client has been authenticated, they were receiving their own thread which they can then communicate with the server.
3. To prevent multiple writes in the same memory location, we will have to implement mutexes or another method that prevents multiple writes at the same time.

## Application Functionality

The application will be a simple task list where any user that is currently authenticated and logged in will be able to edit a universally accessible task list. User’s will be able to do the 4 basic operations (Create, Update, Read, Delete) tasks. If time permits, user will be able to create their own task lists, which will be private by default, but can be accessed by other users through permission levels.

Client Requests

Format:

* (**Header; Body)**
* The header and body will be separated by a colon now, but this could change in the future, if we decide to send advanced data structures instead of a string.
* **Commands**
  + Create
  + Read (Task ID)/ Read
  + Update (Task ID)
  + Delete (Task ID)
* **Header**
  + Client will have a header, displaying what kind of operation the user wants to perform, and the task ID the user wants to edit.
  + Example: (Edit 1), (Create), (Delete 2), (Read 10)
  + For Create operations, no ID will be needed, as the system will automatically handle this.
  + Read operations will print all tasks if no ID is provided, and if an ID is specified, will send the specified task only.
* Body
  + Body will only be needed for Create, and Update requests, as the user will need.
  + Body will contain a maximum of 200 characters, including spaces, and the client program will be able to catch if the size has become too big.
  + Each body will end in “\n\n” to prevent streaming errors, such as when the server receives only partial message. If the server does not receive the ending sequence, the server will then keep reading the string, waiting for more data.

Server Responses:

* Header
  + Header response will replicate HTTP status code, so 200 is a Success, 400 are Client-Side Errors
* Body:
  + The response body will only provide a short description of the error message, such as if the ID of a message doesn’t exit.

## Session Termination

The application continuously prompts the user for commands. If the user types “Exit” then application will then automatically terminate the connection and close the program. The session application will also be terminated once Ctrl + C has been typed as well. The respective application will have a exit message before they are terminated.