3. A document (.pdf or Word) providing detailed protocol design and implementation i.e.

messages/requests being sent by the client and corresponding responses from the

server. Provide format details of both request and response.

**Protocol Design and Implementation**

# Feature:

## Client features:

* Add memos to the server to list.
* View existing memos.
* Delete existing memos.
* Login using an ID.
* Logout.

## Server Features:

* Create, delete, retrieve memos from client messages.
* Time out the user after a set amount of time has passed without communication.

# Implementation:

## Protocol:

### Note request format:

### {“name”: NOTE\_NAME, “note”: NOTE\_DESCRIPTION}

Where:

NOTE\_NAME can be “ALL” or a string.

NOTE\_DESCRIPTION can be a string.

### Note response format:

### {“id”: ID, “name”: NOTE\_NAME, “note”: NOTE\_DESCRIPTION}

Where:

ID numeric identification of the note that is defined by server.

NOTE\_NAME can be “ALL” or a string.

NOTE\_DESCRIPTION can be a string.

### Request format:

{“action”: ACTION, “parameter”: PARAMETER, “token”: TOKEN}

Where:

ACTION can be one in this set ('LOGIN', 'LOGOUT', 'ADD', 'RETRIEVE', 'DELETE')

PARAMETER is a defined IDENTIFIER ("tommy1", "hally2", "pattrick3") if ACTION is “LOGIN”, or it can be “” if the ACTION is “LOGOUT”, or it can be a note request if ACTION is “ADD”, or it can be “ALL”/NOTE\_NAME if ACTION is “RETRIEVE”, or it can be NOTE\_NAME if ACTION is “DELETE”.

TOKEN is an empty string “” if the ACTION is “LOGIN” or it is a number sent by server after logging in for other ACTION.

### Response format:

{“status”: STATUS, “message”: MESSAGE}

Where:

STATUS can be either “SUCCESS” or “ERROR”.

MESSAGE can be an ERROR MESSAGE string if STATUS is ERROR, or it can be a TOKEN if ACTION is “LOGIN”, or it can be an ERROR MESSAGE string for “LOGOUT” or “DELETE”, or it can be a note response or a list of note response for “RETRIEVE”.

# Description

Both the client and server utilize tcp and the socket library. Additional external libraries include time and json and sqlite.

The server binds a port which is set in the source code and listens for the incoming client connections. Once a client connects, the server waits for the LOGIN action from the client. If the client does not send that action, or the server does not recognize the ID sent by the client, the server rejects the client's message and sends back an error. The client will then re-attempt log in using the corrrect ID or terminate the connection. Once the login is successful, the server generates and stores a session token, which the client is given and also stores it. The bulk of the interactions between the client and server then begin.

The client can send a message to the server with any of the available actions, except for LOGIN. The course of action is determined using a match case statement on the client and server.

During an active connection, if there is no activity for a certain amount of time, then the server times out the client connection by removing the token that connection uses, stops accepting data from the client and tells the client to log out.

If the action is ADD, the user provides a memo name and its contents along with TOKEN, which are sent to the server. This action only fails if the new note name is a duplicate of the one in the server or if either the memo title or its description is missing.

If the action is DELETE, the user is asked which memo to delete based on its name along with TOKEN. The server will then delete the associated memo. This action fails if no memo is given, or the memo name is not in the list of memos.

If the action is RETRIEVE, the user is asked which memo to view along with TOKEN. The server will then return the contents of the associated memo, if it exists. This action fails is the memo name given does not exist in the server. Also, by using key word “ALL” for PARAMETER, all the list of memos will be returned.

If the action is LOGOUT, the client will send the session TOKEN it is using to the server. The server will then delete the token, close the incoming client socket, and the client will disconnect from the server. This action should not fail under normal circumstances.

In addition, this protocol has the byte streaming of TCP protocol in the server as it can accept byte by byte from the client as long as the last byte of the message comes in the allowed duration.

In addition, there are many edge cases are handles as below:

Error Cases:

1. Client sends wrong invalid action

Server returns an error message such as "INVALID TOKEN", "INVALID DATA", "INVALID IDENTIFIER", and “OPERATION FAILED”, “EMPTY”, and “IDENTIFIER IN USE”.

The client receives the message, displays it to the user and re-prompts an action.

2. Client sends invalid protocol message

Client does not send a protocol message that does not match with the given format.

The server returns an error message "INVALID PROTOCOL" and the client has to re-enter the message

3. Client connection is closed improperly

On netcat, if the client closes connection the server will keep receiving messages infinitely.

The server will check for a timeout or after certain number of duplicate messages, and then forcibly close the client socket, freeing up the connection.

4. Session timeout

When the session is time out and the client tries to perform an ACTION, the server will return error message with content “SESSION EXPIRED”.