INTRODUCTION

The third and final project of the networking lab will be to create a simple network-based (client/server) **LIST MANAGER** using the knowledge and techniques we've gained from the previous two projects.

BACKGROUND

A **LIST MANAGER** maintains lists like to-do lists, assignment lists, and the like, accessible through a custom client.

The concepts that will be introduced in this project will be:

- Parsing client requests, acting on those requests, and responding appropriately
- Server-side condition and prerequisite checking
- Client-side error detection
- Client-side processing of server responses

In Project 3, we will further extend the functionality and capabilities of the custom client and server applications developed in Project 2.

BEFORE STARTING

Before you jump into this project, please consider making a backup of your Project 2 source code just in case something would go wrong. Also consider using a software repository, like GitHub, to store your source code and organize your projects.

REQUIREMENTS – CLIENT APPLICATION

Create a standalone, interactive, and fully commented client application that interacts with a server application and performs these tasks:

1. Use information from a configuration file to open a log file and write all major activity to the log file
2. Use information from a configuration file to connect to the **LIST MANAGER** server application over the TCP/IP network
3. Loop while prompting the application user for commands. If an empty string was entered, display and log an appropriate error. The list of required commands are:

a.	add <list item>	- Adds an item to the current list
b.	create <list>	- Creates a new list
c.	delete <list>	- Deletes a list
d.	help	- Displays a message showing all available commands
e.	quit	- Gracefully shuts down both server and client applications
f.	remove <list item>	- Removes an item from the current list
g.	show	- Displays a numbered list of the list items

4. Process each input string as follows:
    a. Write the original command string to the log file
    b. Parse the command string into the ***request*** and ***parameter*** parts (as in Project 2) and converting the request to all upper case
    c. Verify that the ***request*** is a valid command. Display and log an appropriate warning message if not
    d. Only valid commands should be sent to the server
5. Create a ***server request*** from the ***request*** and the ***parameter*** as a JSON-formatted string
6. Send the ***server request*** to the server application and wait for a ***server response***
7. Upon receiving a server response, process as follows:
    a. Log the response as received from the server
    b. If the response was to a SHOW command, display the response parameter as a numbered list
    c. If the response was either a WARNING or ERROR, display both the response and parameter
    d. If the response was a QUIT, then display an appropriate message and exit the client application
    e. If the response was anything else, display the parameter

CLIENT APPLICATION DETAILS

1. Copy the Project2 client configuration file to create a client configuration file for Project 3
2. Change the logFile option to indicate a new server log:

   ```
   logFile = <username.#>-project3-client.log
   ```

3. If a HELP command is entered, you must display a "usage" message with all commands. HELP is a client-side only command

   For example:

   ```
   usage:

     add <item>          - Add list item

       create <list name>  - Create list
       delete <list name>  - Delete list
       help                - Help
       quit                - Quit
       remove <item>       - Remove list item
       show                - Show items
   ```

4. If an invalid command is received, you must display and log an appropriate error message stating that an invalid command was received and then display a usage message

   For example:

   ```
   Invalid command entered: xxxx

     usage:

       add <item>          - Add list item
       create <list name>  - Create list
       delete <list name>  - Delete list
       help                - Help
       quit                - Quit
       remove <item>       - Remove list item
       show                - Show items
   ```

5. If a valid command was received but was improperly formatted, you must display and log an appropriate error message and then display a usage message

   For example:

   ```
   Missing element in command: ADD
   ```

```
usage:

  add <item>           - Add list item
  create <list name>   - Create list
  delete <list name>   - Delete list
  help                 - Help
  quit                 - Quit
  remove <item>        - Remove list item
  show                 - Show items
```

REQUIREMENTS – SERVER APPLICATION

Create a standalone and fully commented server application that performs these tasks:

1. Use information from a configuration file to open a log file and write all major activity to the log file
2. Use information from a configuration file to open a socket and bind to the IP address and port
3. If a list from the last session was not deleted then initialize list from the saved list
4. Listen for an incoming client connection. When one arrives, accept it
5. Processing all client requests as follows:

   a. Log the client request as received
   b. If the client request is “ADD”, a list has been created, and the list item is not in the list, add the item to the list and format and send an appropriate server response
   c. If the client request is “CREATE” and a list has not been created, initialize a new list using the client request parameter as its name and format and send an appropriate server response
   d. If the client request is “DELETE” and a list has already created, then reset the list name and format and send an appropriate server response
   e. If the client request is “QUIT”:
      i. If a list has been created, but not deleted, save the list for next application start
      ii. Format and send a server response stating the server is closing
      iii. Close the client connection and exit the application gracefully
   f. If the client request is “REMOVE”, a list has been created, and the list item is in the list, remove item from list and format and send an appropriate server response
   g. If the client request is “SHOW”, a list has been created, and there are item(s) in the list, create a comma separated string of all items in the list, format a server response with the item string in the server response parameter and format and send an appropriate server response
   h. If the client request is anything else, format and send an ERROR server response stating the request is not supported

SERVER APPLICATION DETAILS

1. Copy the Project 2 server configuration file to create a server configuration file for Project 3
2. Change the logFile option to indicate a new server log:

```
logFile = <username.#>-project3-server.log
```

GRADING

To earn credit for this project, you will need to;

1. Record a video of you running your completed project, following the ***Testing Steps*** below, in the OSU Student Linux environment
2. Submit the video, along with your source code and the log files produced after running your server and client applications, to Carmen

TESTING STEPS

1. Login to the OSU Student Linux environment
2. Upload your application (Python script) and the provided configuration file
3. Begin recording
4. Open two sessions to your Student Linux virtual machine
5. Run your server application in one session
6. Run you client application in the other session
7. Execute this testing command sequence (sequence 1) in your client application:

    a. `help`
    b. `crate`
    c. `create`
    d. `remove item`
    e. `create todo`
    f. `create another`
    g. `show`
    h. `add`
    i. `add study for final`
    j. `add take final`
    k. `add party`
    l. `add get grades`
    m. `show`
    n. `remove party`
    o. `show`
    p. `add resolve to study harder`
    q. `remove study for final`
    r. `show`
    s. `quit`

8. Wait 1 minute and restart your server, then your client applications

9. Execute this testing command sequence (sequence 2) in your client application:

    a. `create mylist`
    b. `show`
    c. `remove take final`
    d. `show`
    e. `delete`
    f. `delete mylist`
    g. `delete todo`
    h. `create`
    i. `Add item`
    j. `create StoryOutline`
    k. `add Boy meets girl`
    l. `add Boy and girl fall in love`
    m. `add Earth explodes`
    n. `Show`
    o. `ADD The end`
    p. `show`
    q. `remove earth explodes`
    r. `show`
    s. `remove Earth explodes`
    t. `show`
    u. `delete storyoutline`
    v. `delete StoryOutline`
    w. `quit`

10. Stop recording
11. Download the log files from your client and server applications
12. Create a ZIP file with your source code, your log files, and the recording of your session
13. Submit your ZIP file to Carmen

CREDIT

Credit will be given as follows:

Client Application	
• Client application runs without failures/errors	10 Points
• Client application fulfills each of the stated requirements	10 Points
Server Application	
• Server application runs without failures/errors	10 points
• Server application fulfills each of the stated requirements	15 Points

Partial credit will be given wherever possible.

NOTES

- In standardized UNIX/Linux documentation, the use of the '<' and '>' surrounding a variable (i.e. "<username>") indicates a variable that should be replaced with the variable referenced. For example, if you are given the string "<username.#>", you replace this, including the '<' and '>' with your last name and number (e.g. smith.1234).
- What are "major events" regarding logging? Logging is an important aspect of supportable applications. The definition of "major" is intentionally left vague so to provide the developer latitude of what goes into the log. Consider if a server has been running for a month (or more!) and it is found out that a week and a half ago some transaction was not completed properly. In this situation if a log file is used, you could go to the log file and look back to the date a time to see what happened. The "what happened" are the major events. So, in the case of this project, perhaps "major events" would be anything received from the client, any processing that we do server-side, and any response that is sent back to the client. On the client side, it would be similar. Anything sent to the server, any validation errors, and anything received from the server. All exceptional information should also be logged, (e.g. the inability to open a connection or file).
- Please review the Grading section of the syllabus