Contents of submission folder:

1. ps.py
2. cl.py
3. bk1.py
4. Lexicon.txt
5. WriteUp File.docx
6. Lexicon.txt in subfolder backup

How to execute Python Files:

1. Open the Anaconda prompt
2. Navigate to the folder where the python files are saved using “cd <path>” and run “python <filename>”

OR

Run “python <filename along with the path>”

Execution:

1. Please run the “bk1.py” in the terminal for backup server.
2. Once the Backup Server process starts running, open another terminal and run the “ps.py” file for primary server.
3. Once the Primary Server process starts running, open another terminal and run the “cl.py” file.
4. For each client, please open a new terminal and run “cl.py” file.

Functioning:

cl.py:

1. Initially the user will be asked to enter username.
2. If the username is unique then it will ask the user to select the user-specified input file. If not then it will inform the user that “Username is taken and is active”. Waiting for the user to enter new username.
3. Once the unique username client is connected to the server then it will ask to select the user-specified input file. Along with Browse File, Upload File and Quit buttons. And to add words to lexicon file with entry field and submit button.
4. On Clicking the Browse File button, a new file explorer window will open, using which the file can be selected. Once the file is selected, click on open in the new file explorer window button. Then the new file explorer window opened will close.
5. Then it will update the entry field with the file selected. And a message that which file is selected will be displayed after quit button.
6. On Clicking on Upload File button, the file will be uploaded. Once the upload is completed then “File is uploaded message will be displayed.
7. After completion of searching for the misspelled words in the user-specified file on referring the lexicon file and creating the new file with the format mentioned in the assignment, a message of “<returned file> file with updated results is created in the current directory” will be displayed. From where the client will copy it.
8. On entering word and clicking on submit, new word is added to the queue.
9. Queue inputs are displayed on the window. Once the quue is polled the poll response, queue contents retrieved, and words added to lexicon are added.
10. In the next cycle of polling the status messages will get updated at the same position.
11. On clicking on quit button, the client process will be closed and will become inactive. Therefore, it will allow the clients with the inactivated username to be connected to the server.
12. Checks if Primary server is responding or not. When primary server is not responding then it will update the users that it is not responding and tries to switch connection to backup server by reconnecting and sharing its username.

ps.py:

1. On running the program, the primary server process will be created and will start connecting to the backup server waiting for primary server connection.
2. Then another the server process will be created and will start waiting for clients’ connections.
3. The popped-up server window will display number of currently connected clients, quit button, total users and their usernames, active users, their usernames.
4. On request for client connection, the server will check if the user already exist. If exists, then pass the message that “Username is taken and is active”. If not, then client connection will be established.
5. Created two threads that can run in infinte loop for lexicon addition and for spell check, close the client connection.
6. On receiving the file from client, the server passes the message that file is uploaded and will cross check for mis-spelled words using the Lexicon.txt file available in the folder.
7. Then will create a new file with suffix “Upd\_” , with the same format mentioned in the assignment sheet.
8. On Clicking on quit button in the client window, the threads and the sockets opened, connection of that client username will be closed and will change that username as inactive.
9. On Clicking on quit in the server window, the server process will be closed.
10. For every 60 second if any words are added by clients, then they are polled and the duplicates from the contents are removed. Check if exist in lexicon file or not. If not then it will be added. Update the buffer files with the responses of server.
11. When new words are added to the lexicon file, these words will be sent to backup server . So that the local lexicon file can be updated with these new words.

bk1.py:

1. On running the program, the backup server process will be created and will start waiting for primary server connection.
2. Once it is connected then it will display a message that primary server is connected. It will periodically check if the primary server is available or not. In parallel to adding the new words shared by primary server to its lexicon file available in its local.
3. When the primary server is unavailable then, it will display a message that primary server is unavailable and will close the existing process that is used for primary server and backup server communication. Then open a new process with same port number used by primary server and waits for the new client connections and handles all the responsibilities of the primary server.

Assumptions:

General assumptions:

1. Assumed that the Lexicon file needs to be created and added to the program files folder. It will consist of mis-spelled words separated by spaces.
2. As opensource code can be used is stated in the assignment problem sheet, I referred the code that is available in gitcode (<https://github.com/isiddheshrao/Distributed-Systems>). Assumed that it will be fine to refer it.
3. Assumed that result file can be created in the test sub-folder of current directory that consists of programs.
4. Assumed that local files of backup server such as Lexicon file can be maintained in the sub-directory backup.
5. Assumed that backup server will be started first then the primary server will be started. As the primary server will try to connect to backup server to maintain backup, to connect to backup server initially it should be up and running before primary server.

cl.py:

1. Assumed that initially username will be entered and will be passed to check the uniqueness.
2. Assumed that if the username exists and the message that username exits and is active is displayed in the client window then the client will enter some other username and will click on enter button again.
3. Assumed that once the username check is completed then only spell check, lexicon addition, quit functionalities will be used.
4. Assumed that buffer files can be used for client server interactions for lexicon addition.
5. Assumed that the status of poll, contents retrieved, lexicon update can be displayed at the same position that can be periodically updated on receiving response from the server.
6. Assumed that on switching connection to the backup server the client username will not be changed.

ps.py:

1. Assumed that once the quit button is clicked in the client window then that username will become inactive and can be reused by the other users, so it will be removed from the active client usernames list displayed in the window and will be available in the total usernames list alone.
2. Assumed that on clicking on “X” icon in the client window, the username will not become inactive but however the client window will be closed. So that username will be displayed in the active usernames list, thereby not allowing any other clients to use that username.
3. Assumed that the quit button in the server window will not be used if there are any active client connections.
4. Assumed that by uploading of file means, copying the file from that location to the current working directory of server program.
5. Assumed that a new file with “Upd\_” suffix filename will be created in the current working directory as per the format specified and the message with the file details needs to be passed to the client window, so that the client can copy the file from that location.
6. Assumed that primary server to backup server communication can happen on a backup server.

bk1.py:

1. Assumed that if primary server is connected, no additional connections will not be received.
2. Assumed that the local lexicon copy of file will be same as primary server and will be maintained in the sub-folder backup.
3. Assumed that backup server needs to take responsibilities and accept the connections from same port that primary server used to handle.
4. Assumed that for initial communications between primary server and backup server, we can use different port numbers.
5. Assumed that once backup server takes responsibilities as primary server then it will still update the local lexicon file.