ASSIGNMENT_7 REPORT

Functionalities Report:

1. **`activeGroups():`**
    - This function takes an array of client file descriptors, an array of group information, and an index `i` as input.
    - The function iterates over all the groups and checks if the number of members in a group is greater than zero but less than 5.
    - If the above condition is satisfied, it checks if the admin client (whose file descriptor is stored in `client_fds[i]` is a member of that group. If the admin is a member of the group, the group ID is sent to the admin client.
    - The function returns void.
2. **`sendGroup():`**
    - This function takes an array of client structures, an array of client file descriptors, a buffer, an array of group information, and an index `i` as input.
    - The function extracts the group ID and the message from the input buffer and stores them in `Gid` and `message` variables, respectively.
    - It then finds the name of the sender client by comparing the client file descriptor stored in `client_fds[i]` with the file descriptor of all clients in the `clients` array.
    - The function then iterates over all the groups and finds the group with the given group ID. If the group is found, it sends the message to all members of the group, except the sender client.
    - If the group is a broadcast group, it sends an appropriate message to the sender client.
    - The function returns void.
3. **`active():`**
    - This function takes a buffer, an array of client file descriptors, an array of client structures, and an index `i` as input.
    - The function creates a string that lists the names of all active clients and stores it in the input buffer.
    - The function sends the input buffer to the client whose file descriptor is stored in `client_fds[i]`.
    - The function returns void.
4. **`sendMessage():`**
    - This function takes a buffer, an array of client structures, an array of client file descriptors, and an index `i` as input.
    - The function extracts the destination client ID and the message from the input buffer and stores them in `dest_id_str` and `message` variables, respectively.
    - It then finds the file descriptor of the destination client by comparing the destination client ID with the IDs of all clients in the `clients` array.

- If the file descriptor of the destination client is found, it sends the message to the destination client.
- If the destination client is not found, it sends an appropriate message to the sender client.
- The function returns void.

5. **`BroadcastMsg()`:**
    - This function takes an array of client structures, an array of client file descriptors, a buffer, and an index `i` as input.
    - The function creates a string that contains the name of the sender client and the broadcast message and stores it in a variable `m1`.
    - The function sends the string `m1` to all clients, except the sender client.
    - The function returns void.

6. **MakeGroupReq():**

- This function handles the request made by a client to create a new group. The function takes the client's socket file descriptor, an array of client_t structs, a character buffer, an array of groupInfo structs, the index of the client in the client_fds array, and an array of requestInfo structs. The function creates a new group and adds the client as a temporary admin and member of the group.
- The function first sets the total number of clients in the request to 1 and generates a unique group ID using the `generateId()` function. It then finds the ID and details of the client who made the request using the socket file descriptor. A message is then sent to the client informing them that they are now a temporary admin and member of the new group.
- The function then loops through the client IDs provided in the input buffer and checks if the ID matches the client who made the request. If it matches, the client is added to the requestInfo struct as the first member. If the ID does not match, a message is sent to the client with that ID, requesting their response to join or decline the group.
- The client calling this function must add his ID at first so that he can become the admin of that respective group

7. **check():**

- `check()` function takes an array of requested client information, an array of client information, an index `idx` and an integer `i` as arguments. The function checks if the sum of the positive and negative responses for the requested clients equals the total number of clients. If true, the function returns true; otherwise, it returns false.

8. **joinGroup():**

    - `joingroup()` function takes an array of client file descriptors, an array of client information, a character buffer, an array of group information, an integer `i`, and an array of requested client information as arguments. The function extracts the group id from the buffer, and then checks for the existence of the group id in the requested client information array. If the group id exists, the function adds the client to the requested client information array and increments the positive response pointer. If the total number of requested clients for the group equals the sum of the positive and negative response pointers, the function returns true; otherwise, it returns false.

9. **declineGroup():**
    - `declinegroup()` function takes the same arguments as `joingroup()`, but instead of adding the client to the requested client information array, it adds the client to the negative response pointer. If the total number of requested clients for the group equals the sum of the positive and negative response pointers, the function returns true; otherwise, it returns false.

10. **send_to_all_requested_clients():**
    - This function sends a message to all the clients who have requested to join a particular group. It takes in an array of `requestInfo` structures, an array of `groupInfo` structures, an array of client file descriptors, an array of `client_t` structures, an index `i` to keep track of the current client, and a group ID `gid` for which the message is to be sent.
    - The function first searches for the index of the requested group in the `requestInfo` array using the provided `gid`. It then checks if the number of positive responses for the group is greater than or equal to the number of negative responses. If this is true, the function creates a new group in the `groupInfo` array with the `gid`, adds the `admin` of the group to the `admins` array, adds all the positive members to the `members` array, sends a message to all members that they have been added to the group, and increments the `Groupidx` counter. If there are not enough positive responses, it sends a message to all requested clients that the group was not created.

11. **makeGroupBroadcast():**
    - This function sets a particular group as a broadcast group. It takes in an array of client file descriptors, an array of `client_t` structures, a buffer containing the group ID and command, an index `i` to keep track of the current client, and an array of `groupInfo` structures.
    - The function first extracts the `reqID` of the client from the `clients` array using the client file descriptor. It then searches for the index of the group in the `groupInfo` array using the provided `Gi`. If the client is not an admin of the group, it sends a message to the client that they are not authorised to make the group a broadcast group. Otherwise, it sets the `isBroadcast` flag of the group to `1` and sends a message to the client that the group is now a broadcast group.

12. **makeGroup():**

    - The makeGroup function creates a new group by parsing a buffer that contains a list of client IDs. The function uses strtok to split the buffer into separate client IDs, which it then uses to populate a struct groupInfo. The function also assigns an ID to the new group, generates an ID for the group, and assigns the first client in the list as the admin of the group. The function then sends a message to all the clients in the group to inform them that they have been added to the group.
    - The client calling this function must add his ID at first so that he can become the admin of that respective group

13. **AddtoGroup():**

    - The AddtoGroup function allows an admin user to add a new client to an existing group. The function parses a buffer that contains the ID of the group and the ID of the client to be added.
    - The function then searches for the group with the given ID and checks whether the client adding the new member is an admin of the group. If the admin user is not found, the function sends a message to the user indicating that they do not have admin privileges.
    - If the group has reached its maximum capacity of 5 members, the function sends a message to the admin user indicating that the limit has been

exceeded. Otherwise, the function adds the new member to the group and sends a message to the client indicating that they have been added to the group.

14. **RemovefromGroup():**

- This function is used to remove a member from a group. It takes in the buffer, which contains the message sent by the client, the client_fds, which is an array of file descriptors of connected clients, clients, which is an array of client_t structures containing information about each client, Groups, which is an array of groupInfo structures containing information about each group, and i, which is the index of the client in the clients and client_fds arrays.
- The function first extracts the id of the client sending the message from the clients array using the client_fds array.
- It then extracts the id of the group and finds the index of the group in the Groups array. It then checks if the client is an admin of the group, and if not, it sends a message to the client saying they are not an admin.
- If the client is an admin, it then extracts the ids of the clients to be removed from the message and removes them from the group. It then sends a message to each removed client saying they have been removed from the group.

15. **Quit():**

- This function is used to terminate a client's connection. It takes in clients, which is an array of client_t structures containing information about each client, client_fds, which is an array of file descriptors of connected clients, address, which is the address of the client to be terminated, addrlen, which is the length of the address structure, buffer, which contains the message sent by the client, and i, which is the index of the client in the clients and client_fds arrays.
- The function sends a message to the client saying their connection has been terminated, and then closes the client's file descriptor.
- It then sends a notification to all remaining clients that the terminated client has left the chat. Finally, it removes the client from the clients and client_fds arrays.