Multi_User_Messaging

Multithreaded Realtime Console Chat App 💬

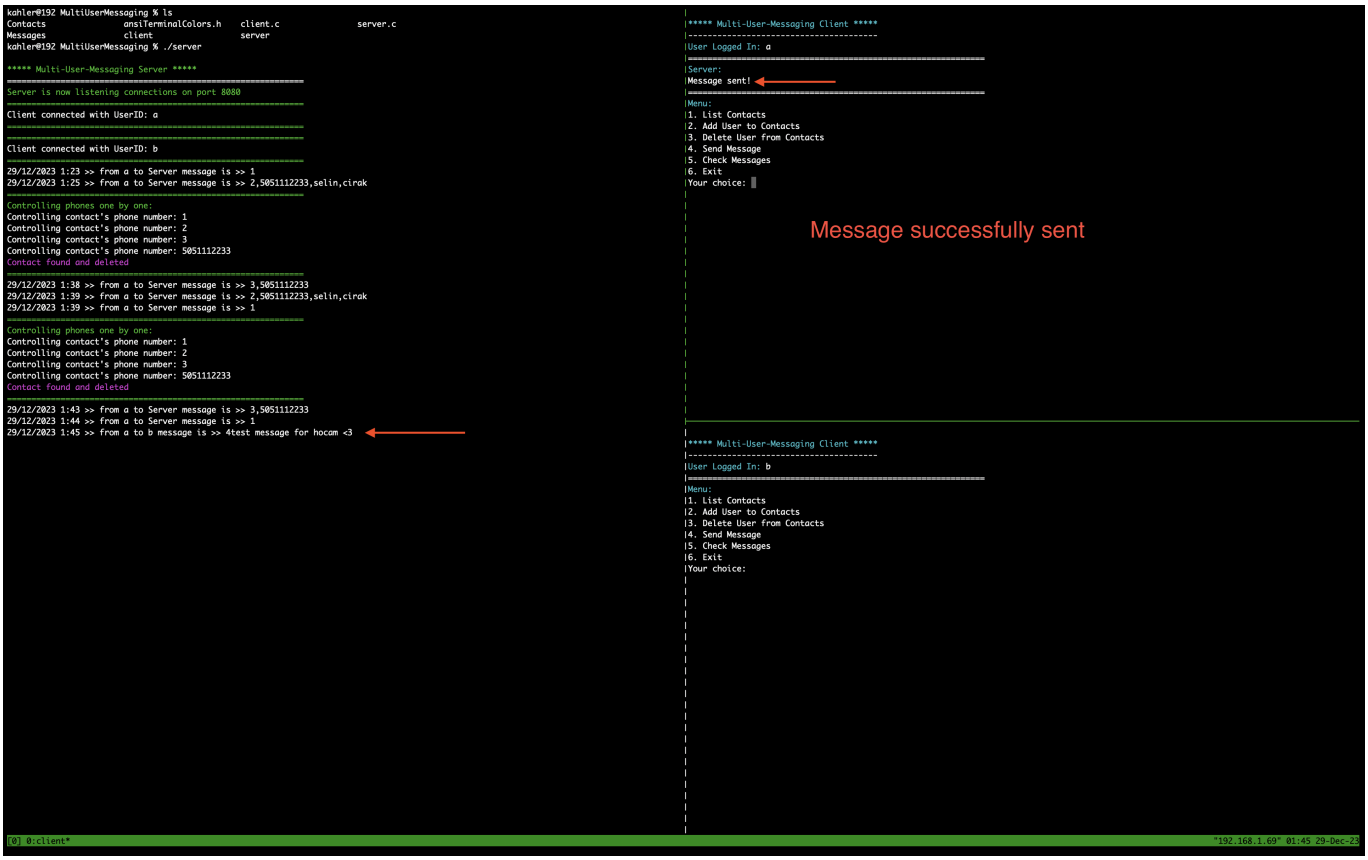


Table of Contents:

Project Description:

> Multi-threaded `console` chat application implemented in C, consisting of both server and client components. The server manages user contacts and messages, while the client provides a user-friendly interface to interact with the server. The communication between the server and client is facilitated through socket programming.

Built With:

```
Programming Languages:
+ C

Frameworks and Technologies:
- No frameworks used (only standard libraries)
+ Only C standard libraries for Unix-like systems
+ Csv file system for storing messages and contacts
```

Libraries and Dependencies:

- Standart libraries Listed Only Important Ones:
  - arpa/inet.h (for socket programming)
  - unistd.h (for read and write functions)
  - pthread.h (for multithreading)
  - time.h (for date and time functions)
  - sys/types.h (for socket programming)
  - sys/stat.h (for file permissions)
- External libraries:
  - ansiTerminalColors.h (written by me)
- Dependencies:
  - gcc (GNU Compiler Collection)

Compatible with:

```
+ Linux
+ MacOS
- Windows (needs to include winsock.h instead of arpa/inet.h)
```

Usage:

It is well described with screenshots in Screenshots section.

Project Structure:

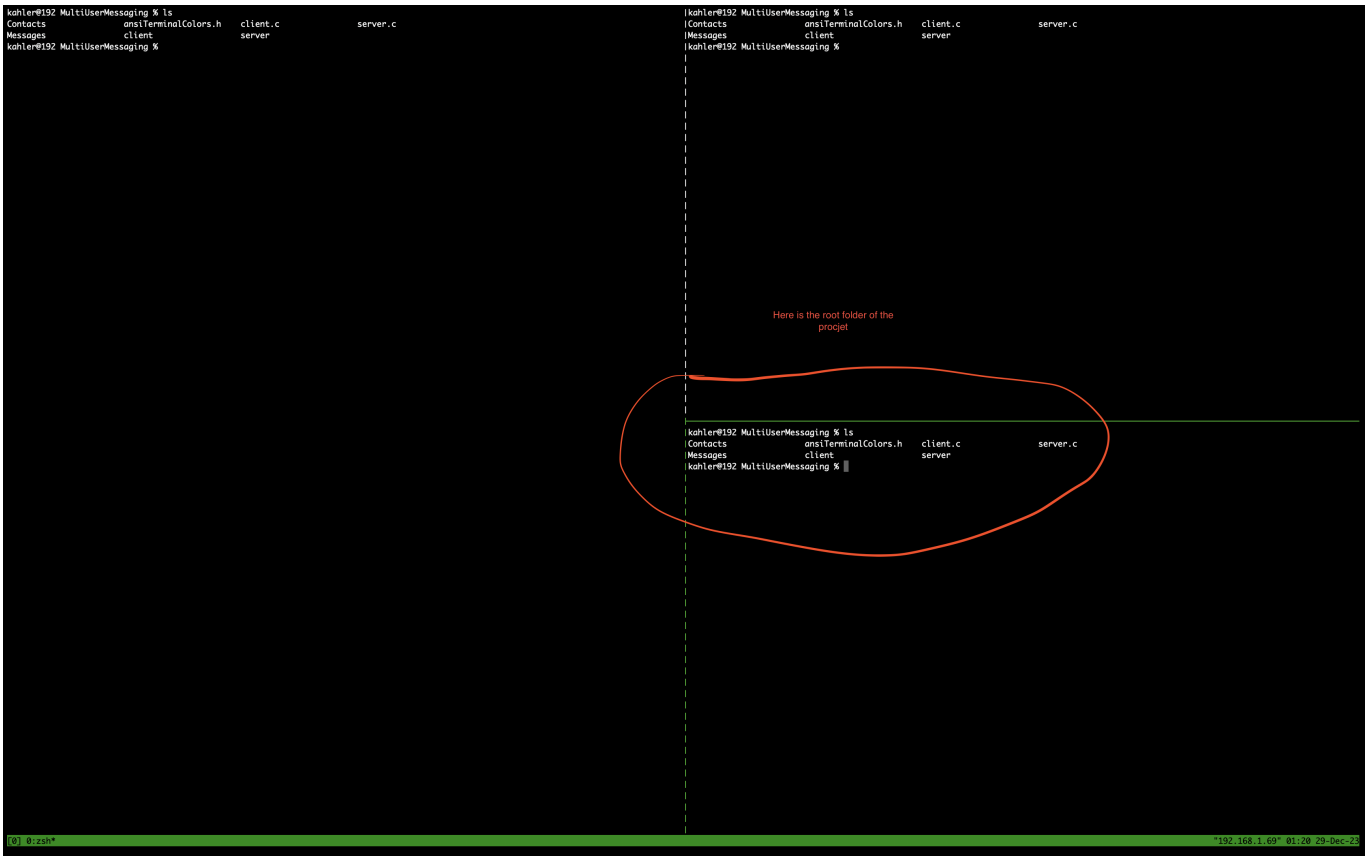
```
.
├── Contacts
│   ├── userID.csv // example
│   └── ...
├── Messages
│   ├── userID.csv // example
│   └── ...
├── Public // this folder is not included in the project
│   └── Project ScreenShots
│       └── ...
├── Readme.md
├── ansiTerminalColors.h // for colored output
├── client // executable of client component
├── client.c // source code of client component
├── server  // executable of server component
└── server.c // source code of server component

5 directories, 24 files
```

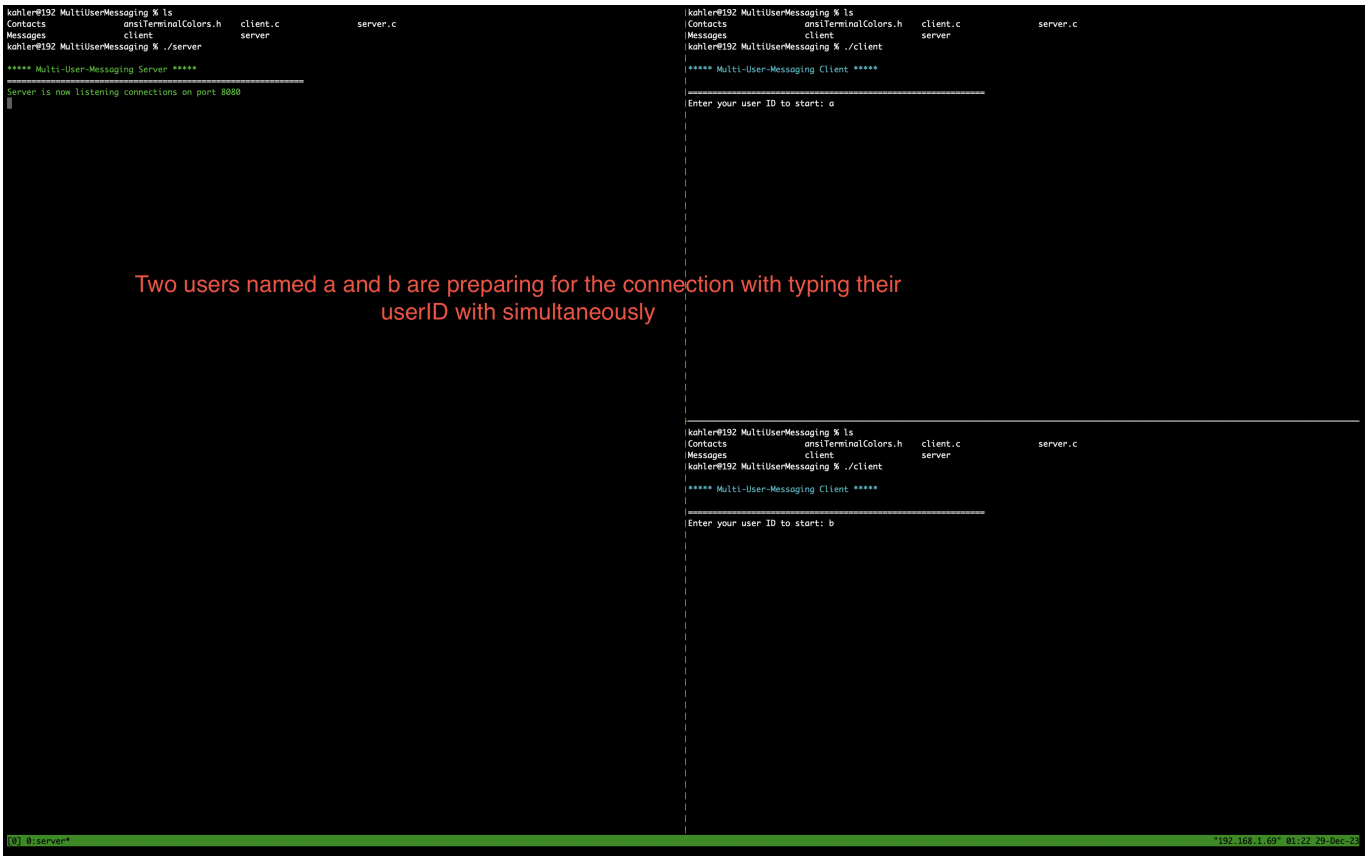
Future Improvements:

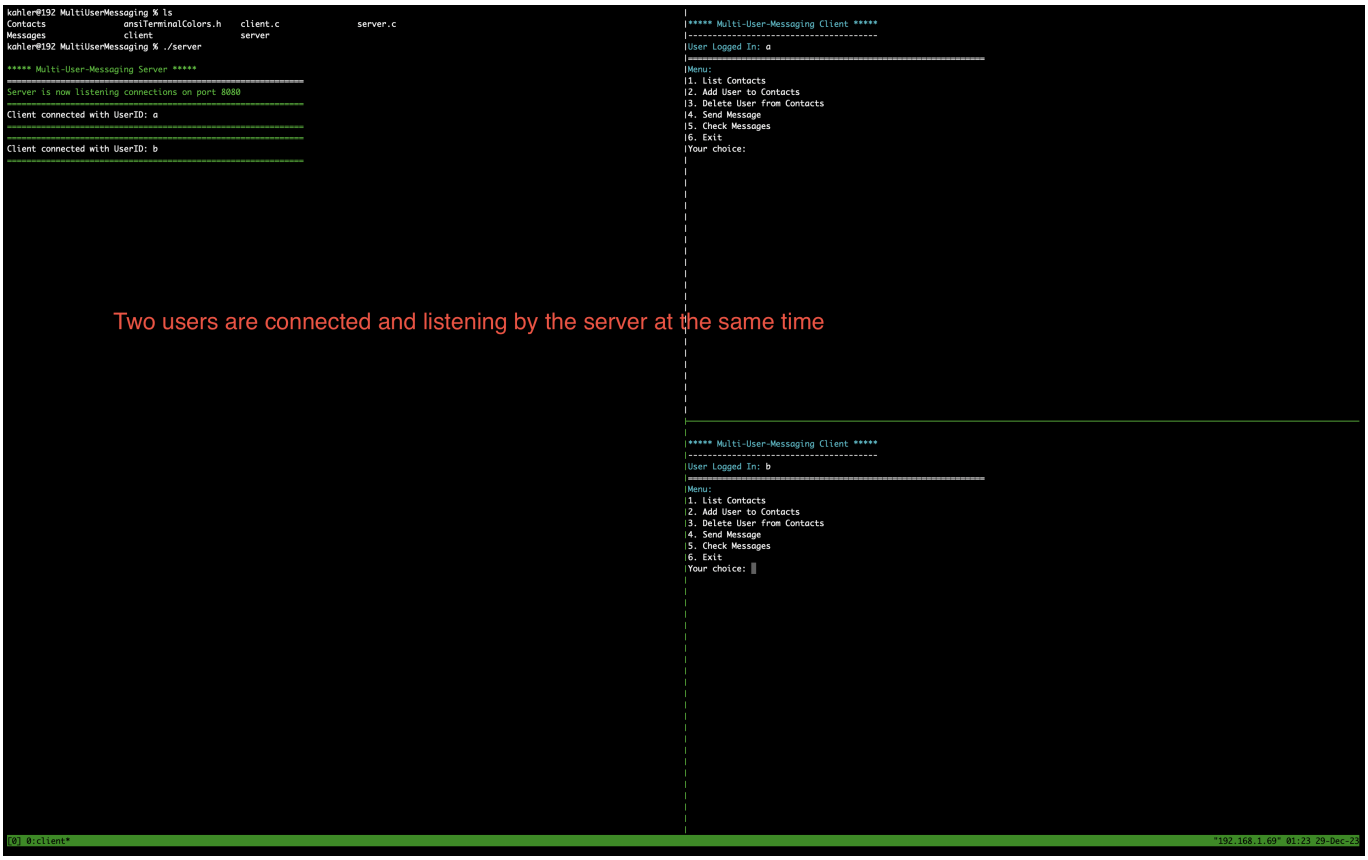
```
+ Add a proper database instead of CSV files (maybe SQLite or PostgreSQL)
+ Add a decent GUI (maybe with Qt)
```

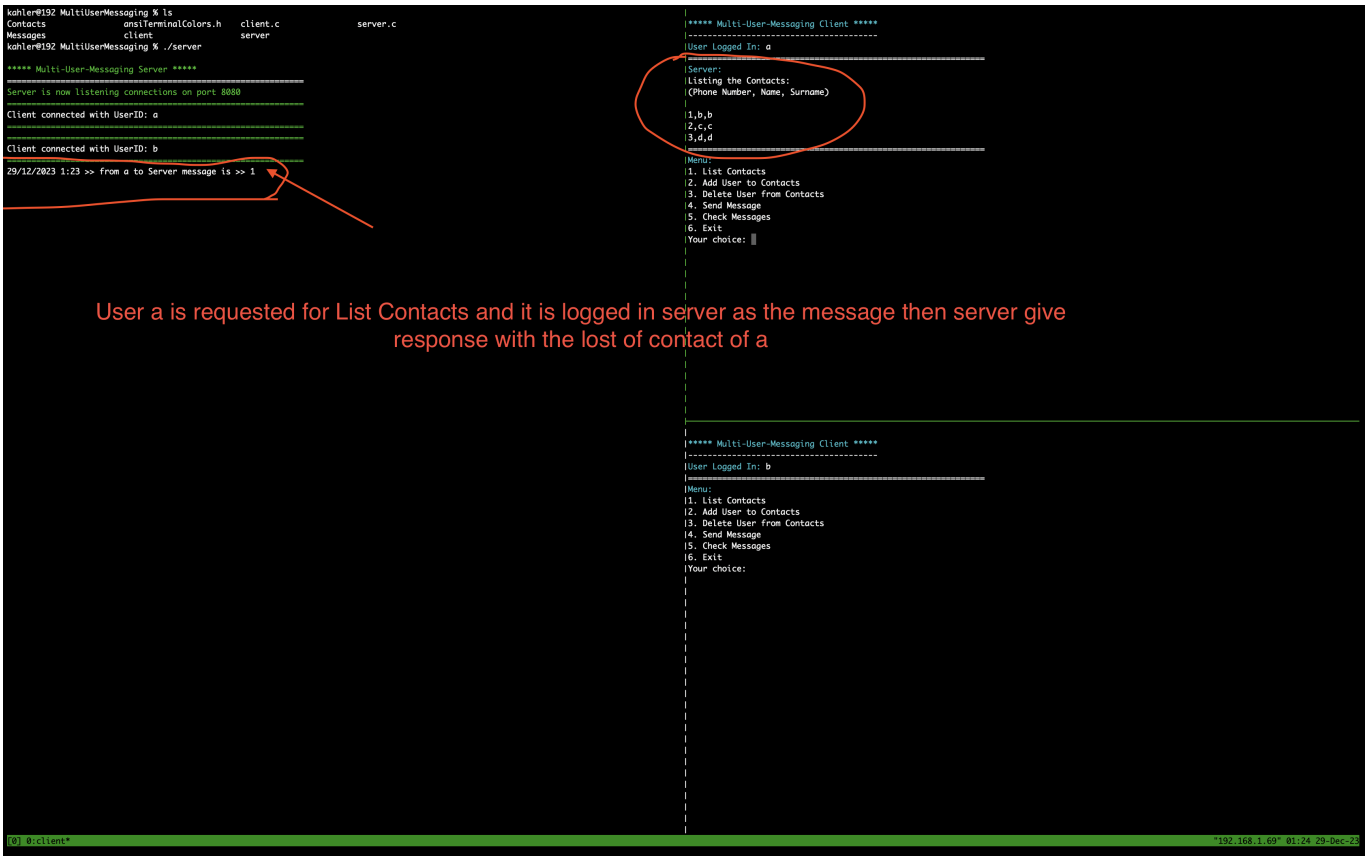
Screenshots:

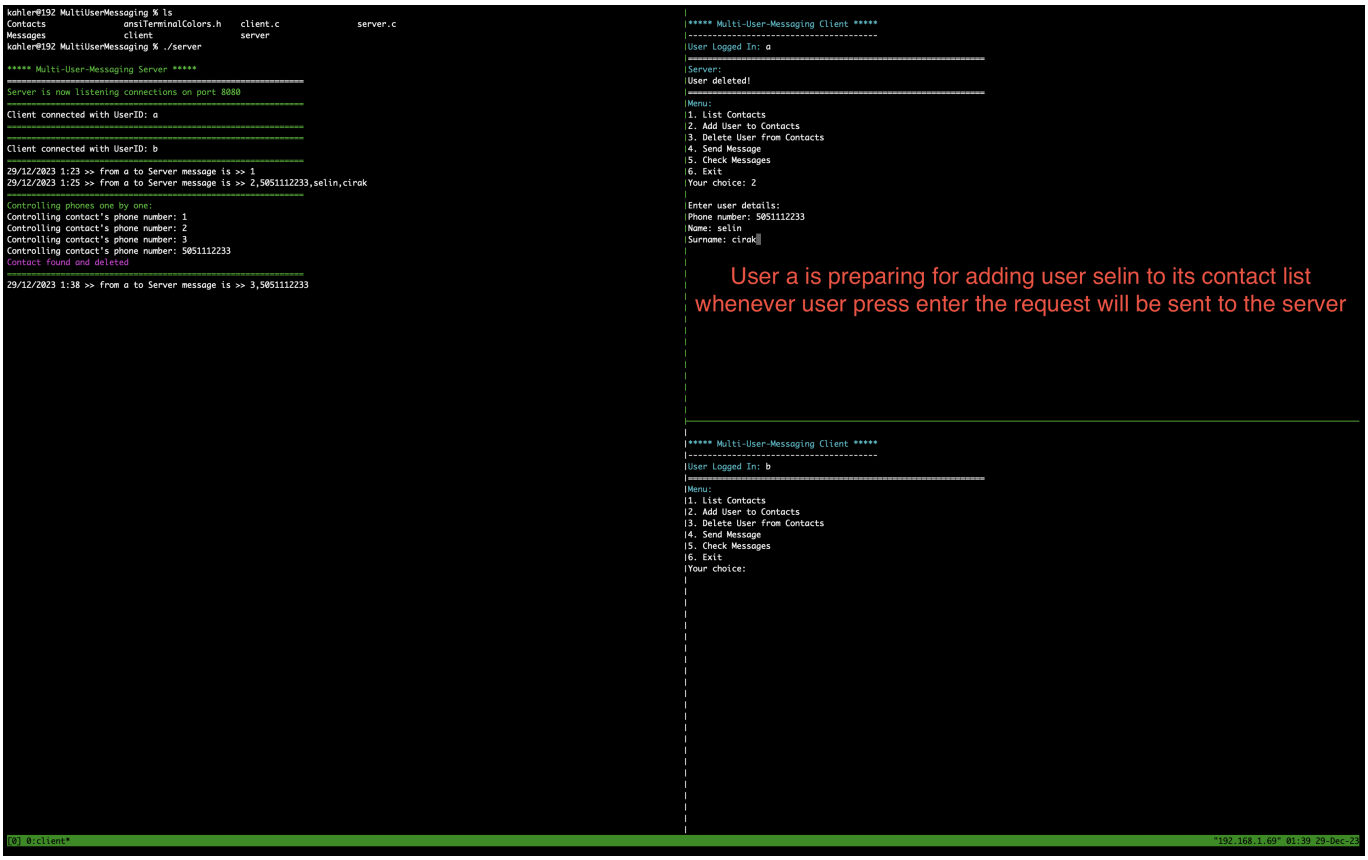

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging %
Here is the root folder of the procjet
[0] 0:zsh*
"192.168.1.69" 01:20 29-Dec-23

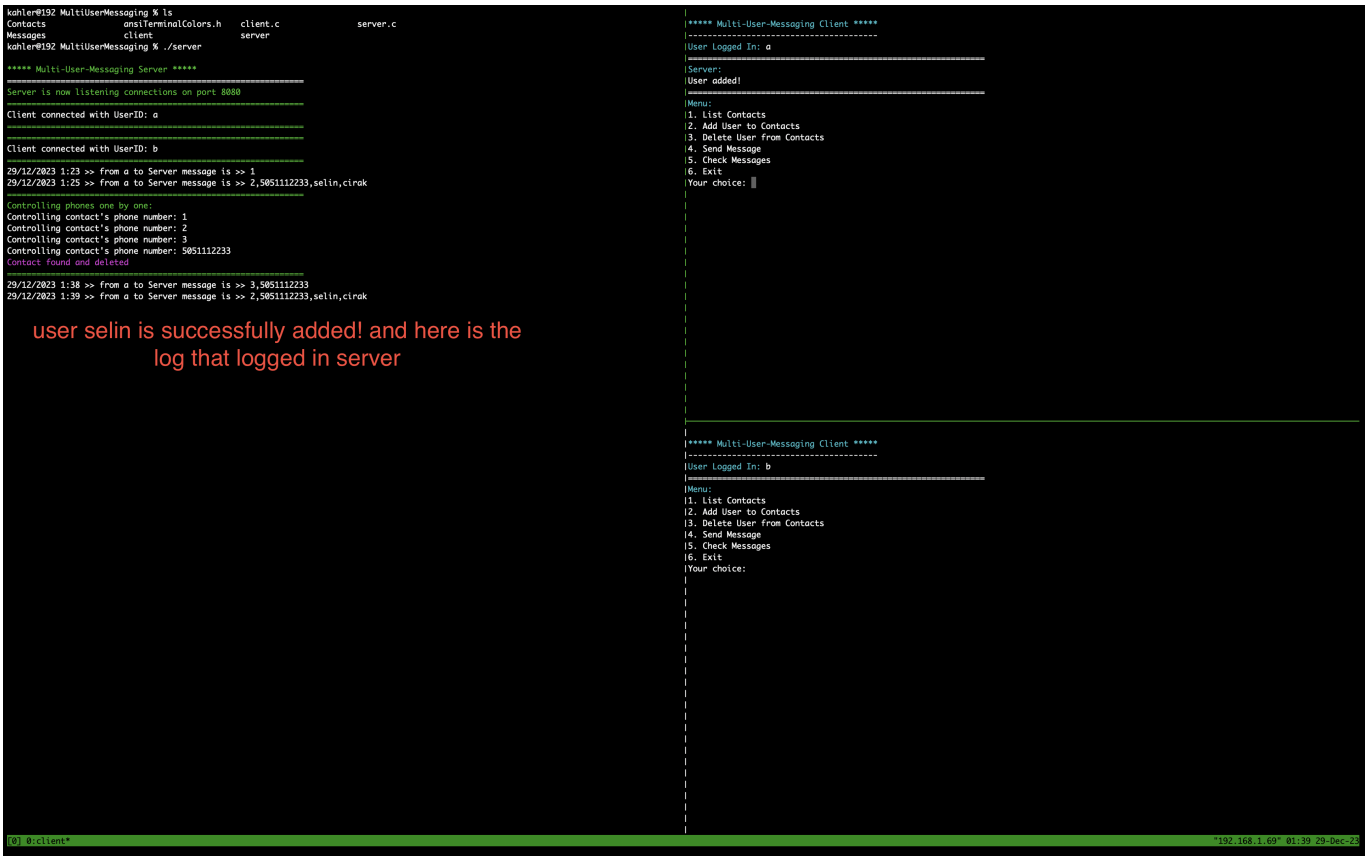

kahler@192 MultiUserMessaging % ls -R
Contacts ansiTerminalColors.h client.c server.c
Messages client server
./Contacts:
a.csv b.csv
./Messages:
a.csv b.csv
All file structure is recursively listed with -R flag

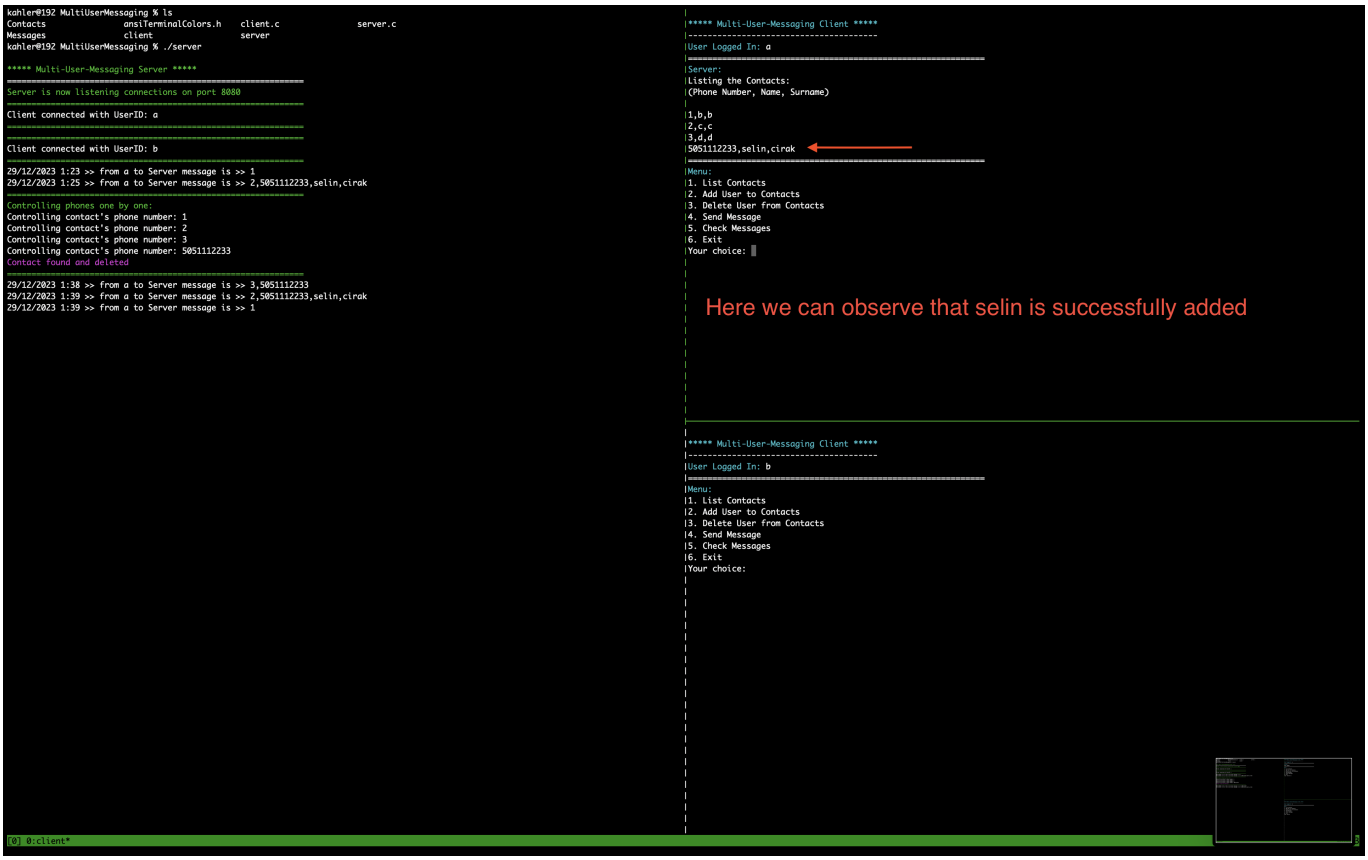

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./client
***** Multi-User-Messaging Client *****
Enter your user ID to start: a
kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./client
***** Multi-User-Messaging Client *****
Enter your user ID to start: b
Two users named a and b are preparing for the connection with typing their userID with simultaneously
[0] 0:server*
"192.168.1.69" 01:22 29-Dec-23

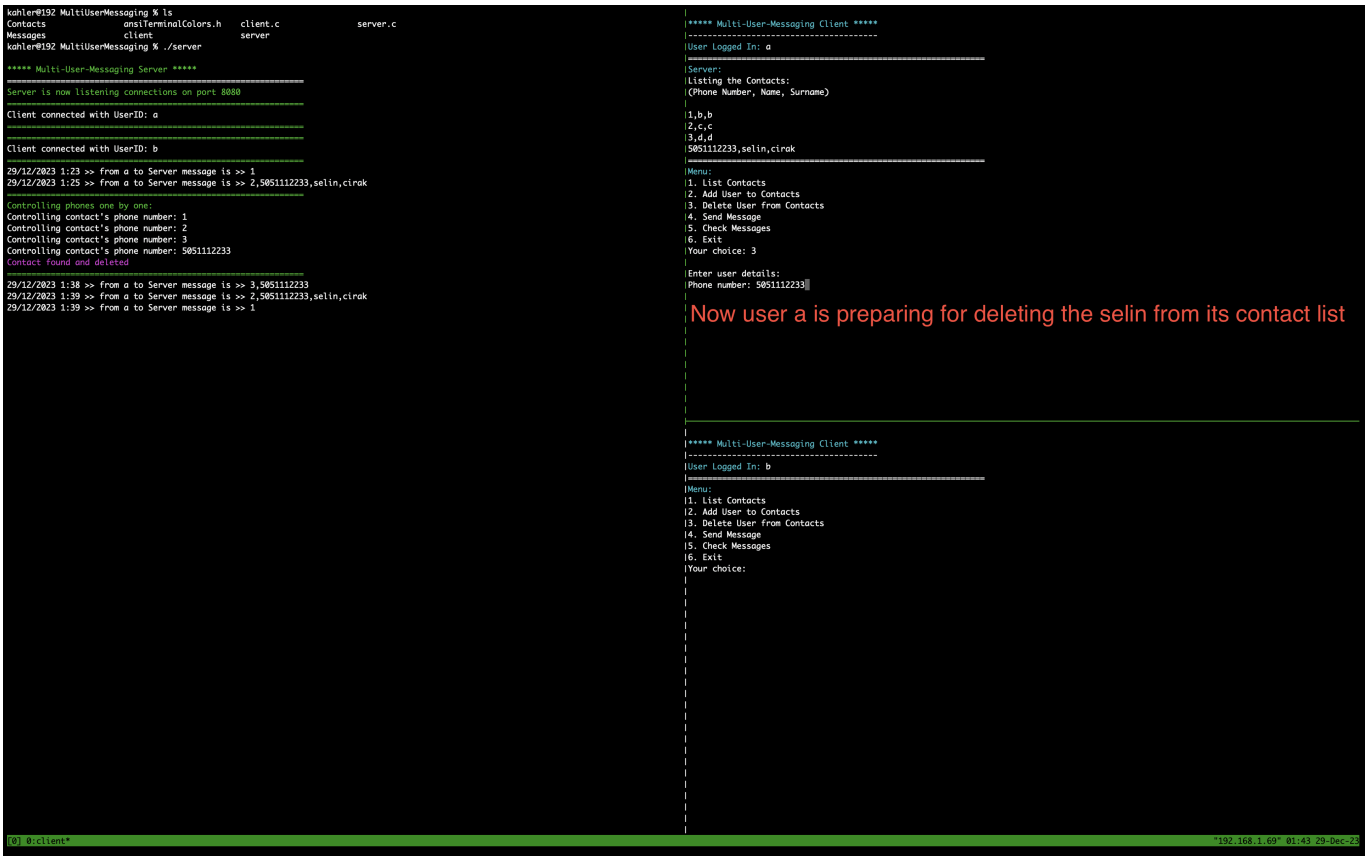

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
***** Multi-User-Messaging Client *****
User Logged In: a
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
Two users are connected and listening by the server at the same time
[0] 0:client*
"192.168.1.69" 01:23 29-Dec-23

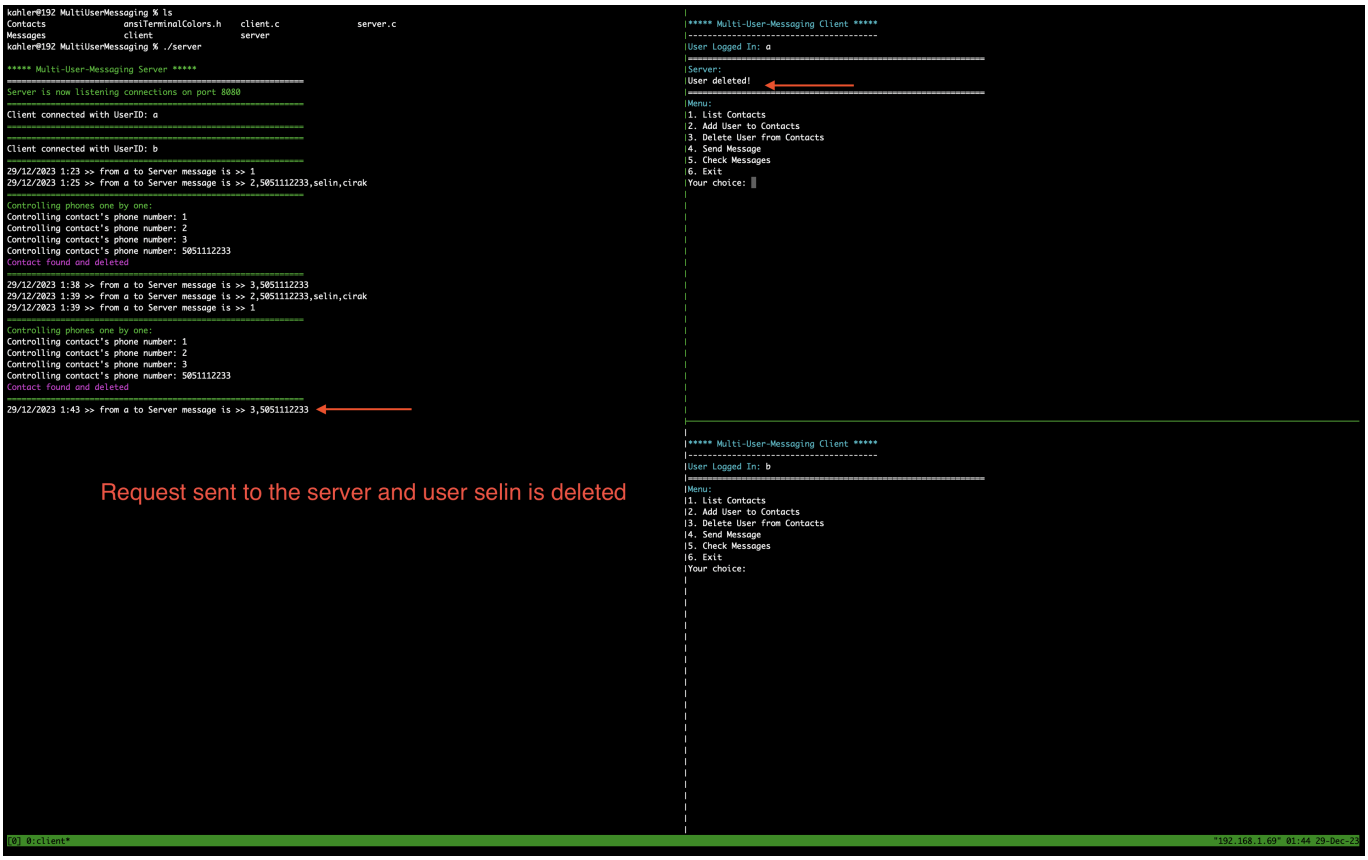

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
Listing the Contacts:
(Phone Number, Name, Surname)
1,b,b
2,c,c
3,d,d
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
User a is requested for List Contacts and it is logged in server as the message then server give response with the lost of contact of a
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:24 29-Dec-23

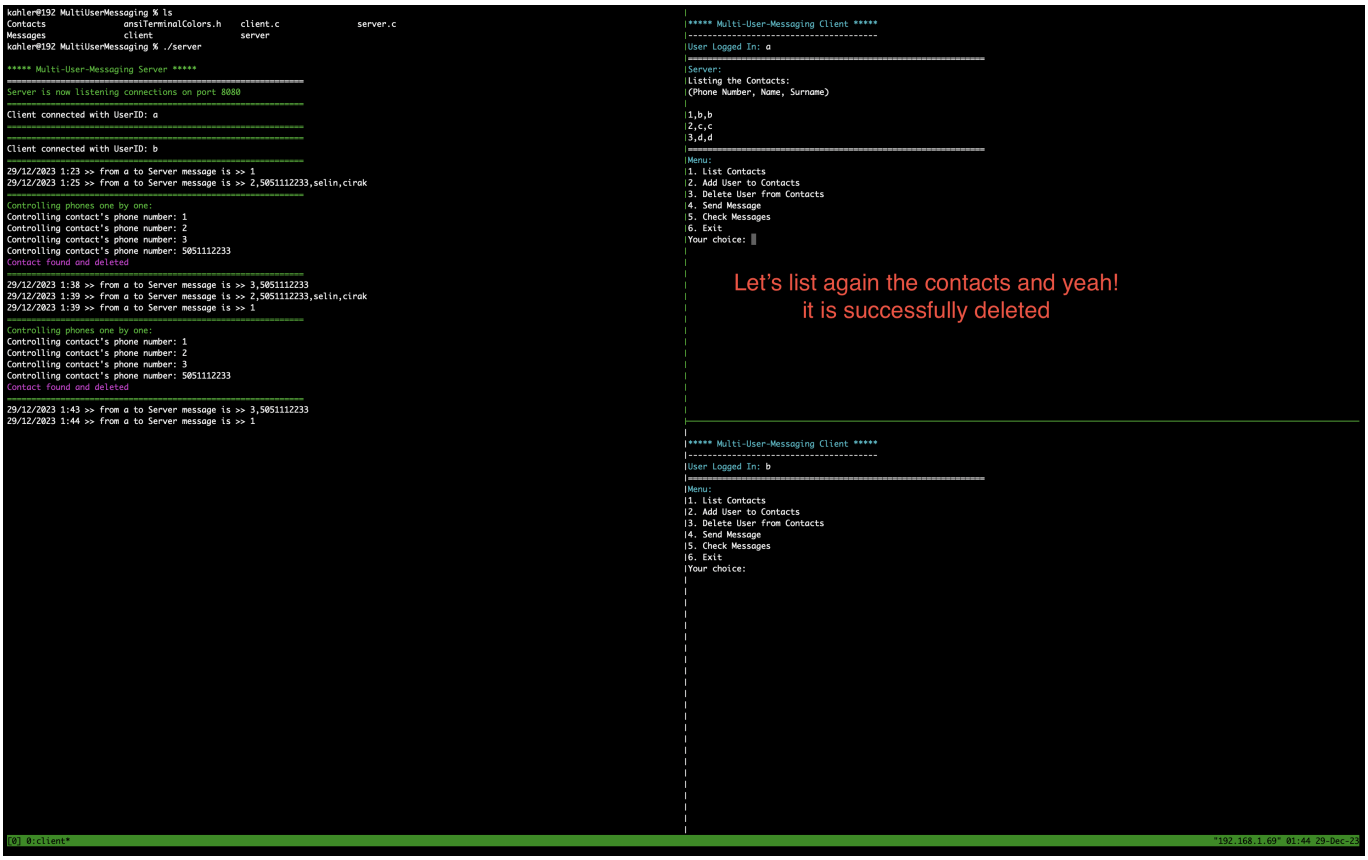

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
User deleted!
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice: 2
Enter user details:
Phone number: 5051112233
Name: selin
Surname: cirak
User a is preparing for adding user selin to its contact list whenever user press enter the request will be sent to the server
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:39 29-Dec-23

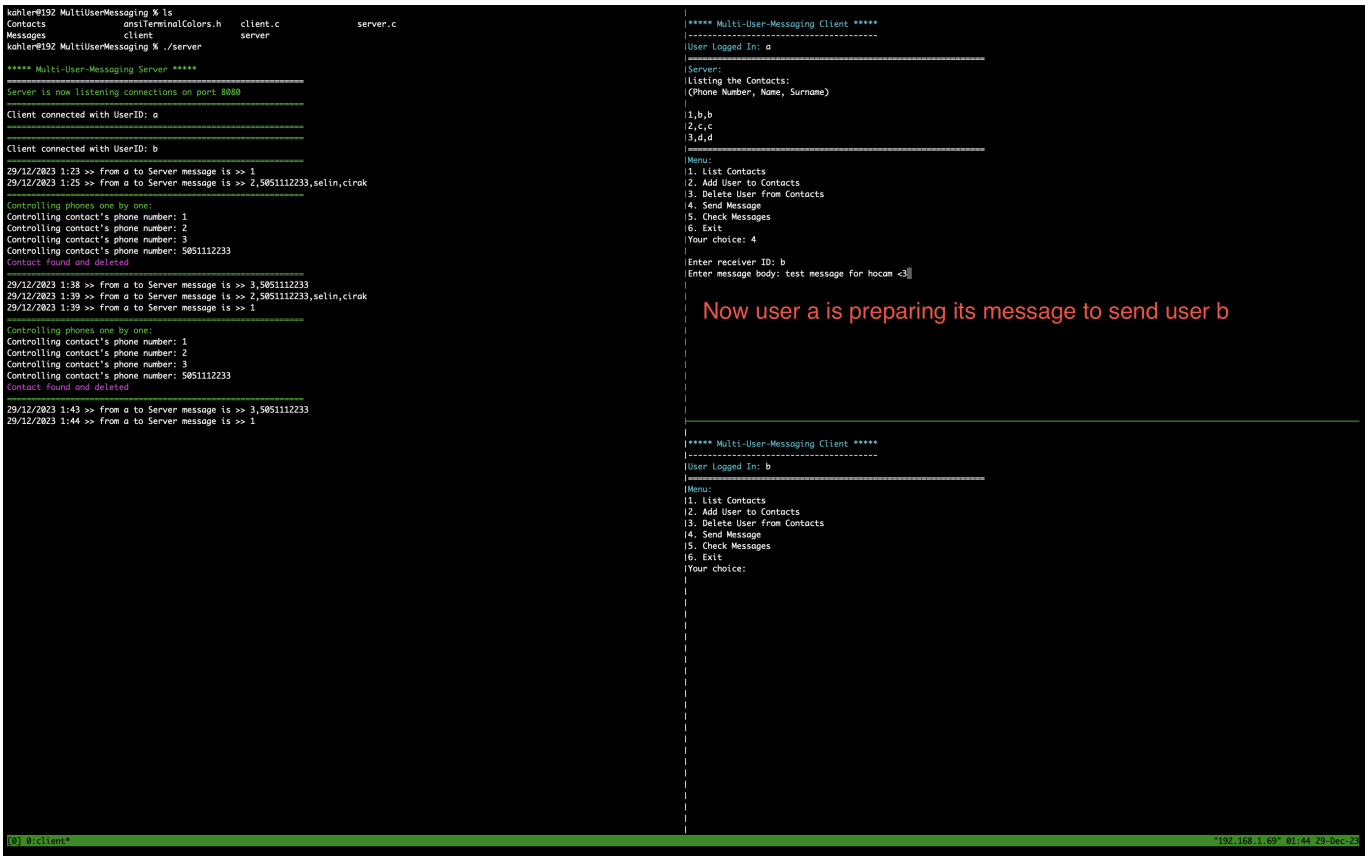

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
user selin is successfully added! and here is the log that logged in server
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
User added!
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:39 29-Dec-23

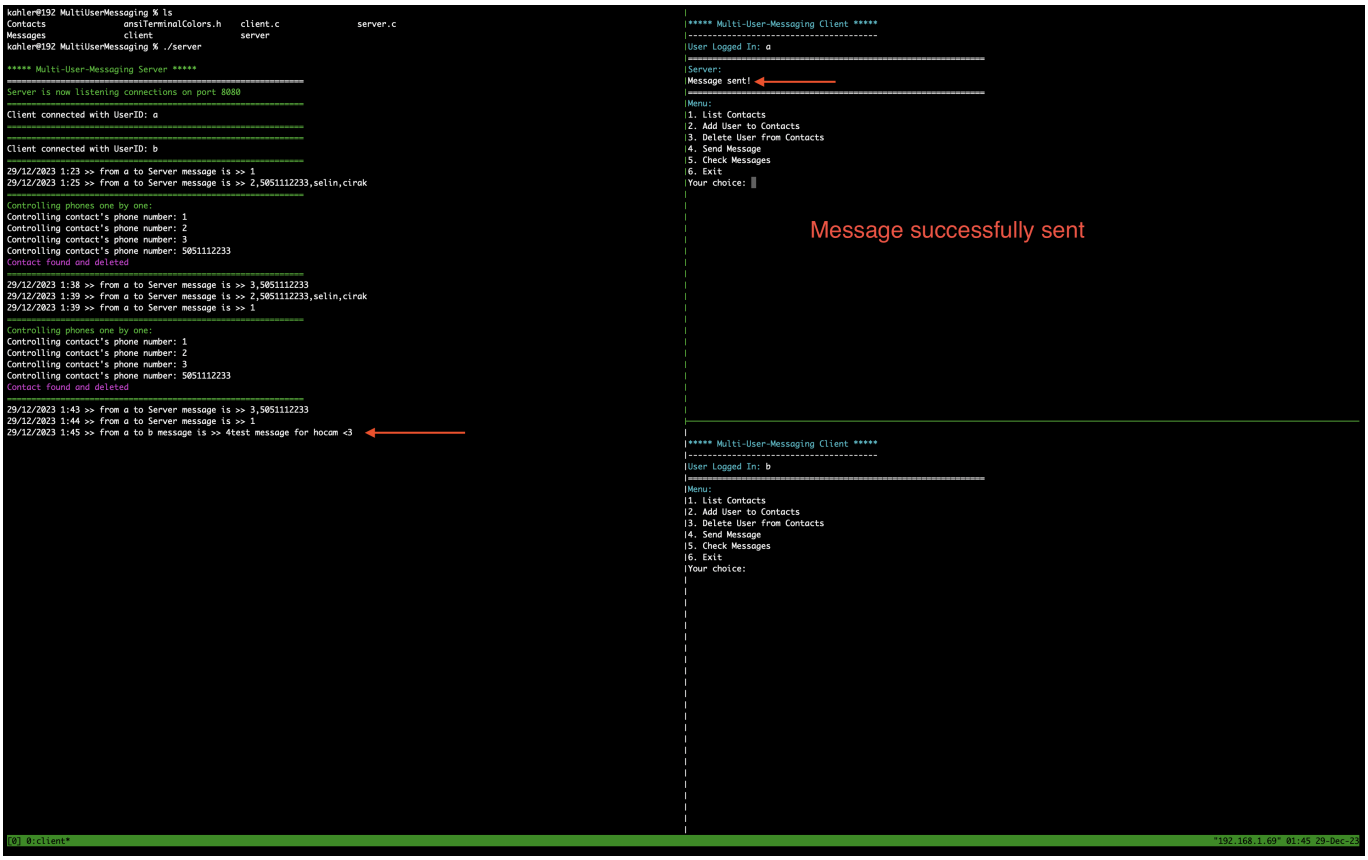

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
29/12/2023 1:39 >> from a to Server message is >> 1
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
Listing the Contacts:
(Phone Number, Name, Surname)
1,b,b
2,c,c
3,d,d
5051112233,selin,cirak
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
Here we can observe that selin is successfully added
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*

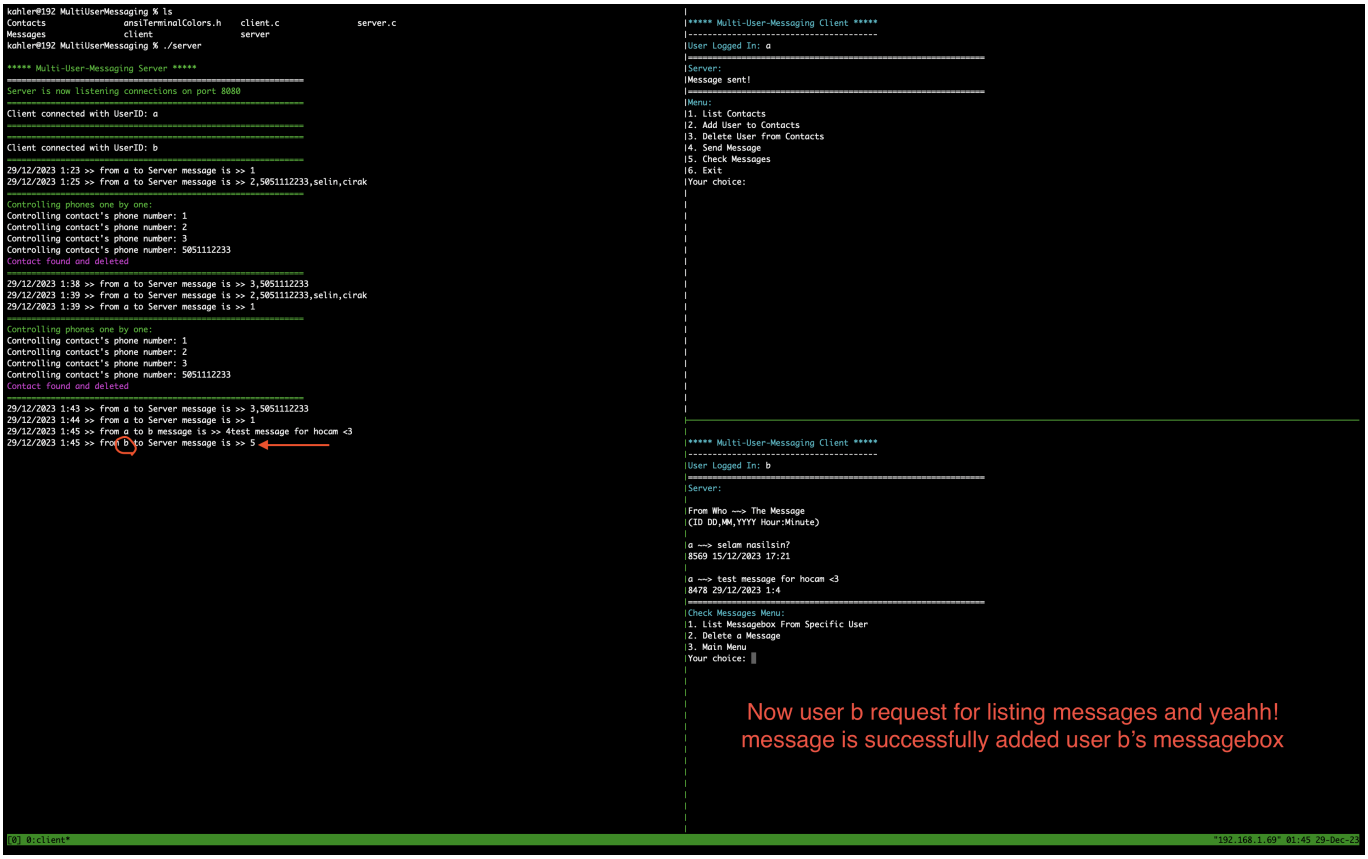

kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
29/12/2023 1:39 >> from a to Server message is >> 1
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
Listing the Contacts:
(Phone Number, Name, Surname)
1,b,b
2,c,c
3,d,d
5051112233,selin,cirak
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice: 3
Enter user details:
Phone number: 5051112233
Now user a is preparing for deleting the selin from its contact list
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:43 29-Dec-23


kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
29/12/2023 1:39 >> from a to Server message is >> 1
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:43 >> from a to Server message is >> 3,5051112233
Request sent to the server and user selin is deleted
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
User deleted!
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:44 29-Dec-23


kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
29/12/2023 1:39 >> from a to Server message is >> 1
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:43 >> from a to Server message is >> 3,5051112233
29/12/2023 1:44 >> from a to Server message is >> 1
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
Listing the Contacts:
(Phone Number, Name, Surname)
1,b,b
2,c,c
3,d,d
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
Let's list again the contacts and yeah!
it is successfully deleted
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:44 29-Dec-23


kahler@192 MultiUserMessaging % ls
Contacts ansiTerminalColors.h client.c server.c
Messages client server
kahler@192 MultiUserMessaging % ./server
***** Multi-User-Messaging Server *****
Server is now listening connections on port 8080
Client connected with UserID: a
Client connected with UserID: b
29/12/2023 1:23 >> from a to Server message is >> 1
29/12/2023 1:25 >> from a to Server message is >> 2,5051112233,selin,cirak
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:38 >> from a to Server message is >> 3,5051112233
29/12/2023 1:39 >> from a to Server message is >> 2,5051112233,selin,cirak
29/12/2023 1:39 >> from a to Server message is >> 1
Controlling phones one by one:
Controlling contact's phone number: 1
Controlling contact's phone number: 2
Controlling contact's phone number: 3
Controlling contact's phone number: 5051112233
Contact found and deleted
29/12/2023 1:43 >> from a to Server message is >> 3,5051112233
29/12/2023 1:44 >> from a to Server message is >> 1
***** Multi-User-Messaging Client *****
User Logged In: a
Server:
Listing the Contacts:
(Phone Number, Name, Surname)
1,b,b
2,c,c
3,d,d
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice: 4
Enter receiver ID: b
Enter message body: test message for hocam <3
Now user a is preparing its message to send user b
***** Multi-User-Messaging Client *****
User Logged In: b
Menu:
1. List Contacts
2. Add User to Contacts
3. Delete User from Contacts
4. Send Message
5. Check Messages
6. Exit
Your choice:
[0] 0:client*
"192.168.1.69" 01:44 29-Dec-23





Code Documentation

Server Component:

Function Prototypes and Explanations:

1. handler function for each client thread

```
void *handleClient(void *arg)
```

> This function runs in a separate thread for each connected client. It handles incoming messages from the client, interprets user input, and performs corresponding actions.

Input	void *arg (client socket)
Output	None

2. Getter function for the current date and time

```
Date getCurrentDateAndTime()
```

> This function retrieves the current date and time and returns it as a `Date` structure.

Input	None
Output	Date (structure)

3. **`void listContacts(char *userID, int client_socket)`**
   - **Inputs:** `char *userID`, `int client_socket`
   - **Outputs:** Sends a list of contacts to the client
   - **Explanation:** Reads the user's contacts from a CSV file and sends them to the client.
4. **`void listMessagesFromUser(char *userID, Message message, int client_socket)`**
   - **Inputs:** `char *userID`, `Message message`, `int client_socket`
   - **Outputs:** Sends messages from a specified user to the client
   - **Explanation:** Reads messages from a CSV file, filters them based on the specified user, and sends the result to the client.
5. **`void deleteMessage(char *userID, Message message, int client_socket)`**
   - **Inputs:** `char *userID`, `Message message`, `int client_socket`
   - **Outputs:** Sends a confirmation or error message to the client
   - **Explanation:** Deletes a specified message from the user's message history and sends a confirmation or error message to the client.
6. **`void addUser(char *userID, Message message, int client_socket)`**
   - **Inputs:** `char *userID`, `Message message`, `int client_socket`

- **Outputs:** Sends a confirmation message to the client
- **Explanation:** Adds a user to the contact list and sends a confirmation message to the client.

7. `void deleteUser(char *userID, Message message, int client_socket)`
   - **Inputs:** `char *userID`, `Message message`, `int client_socket`
   - **Outputs:** Sends a confirmation or error message to the client
   - **Explanation:** Deletes a specified user from the contact list and sends a confirmation or error message to the client.
8. `void sendMessage(char *userID, Message message, int client_socket)`
   - **Inputs:** `char *userID`, `Message message`, `int client_socket`
   - **Outputs:** Sends a confirmation message to the client
   - **Explanation:** Sends a message from the user to another user, updating the recipient's message history.
9. `void checkMessages(char *userID, int client_socket)`
   - **Inputs:** `char *userID`, `int client_socket`
   - **Outputs:** Sends a list of messages to the client
   - **Explanation:** Reads the user's messages from a CSV file and sends them to the client.
10. `void sortTheCSVFileAccordingToDate(char *messagesCSVPath)`
    - **Inputs:** `char *messagesCSVPath`
    - **Outputs:** None
    - **Explanation:** Sorts the messages in a CSV file based on their dates.
11. `int compareDates(const Date *date1, const Date *date2)`
    - **Inputs:** `const Date *date1`, `const Date *date2`
    - **Outputs:** Returns an integer (comparison result)
    - **Explanation:** Compares two date structures and returns the result.
12. `void createCSVIfNotExists(char *userID)`
    - **Inputs:** `char *userID`
    - **Outputs:** None
    - **Explanation:** Creates contacts and messages CSV files if they don't exist for the given user.

Client Component:

Function Prototypes and Explanations:

1. `void displayMenu()`
   - **Inputs:** None
   - **Outputs:** None
   - **Explanation:** Displays the menu options for the user.
2. `void sendMessageToServer(int client_socket, char *userID)`

- **Inputs:** `int client_socket`, `char *userID`
- **Outputs:** Sends a message to the server
- **Explanation:** Takes user input for sending a message, constructs a `Message` structure, and sends it to the server.

3. **`void receiveMessagesFromServer(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Displays messages received from the server
   - **Explanation:** Receives and displays messages from the server.
4. **`void addUserToContacts(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Adds a user to the contacts list
   - **Explanation:** Takes user input for adding a contact, constructs a `Message` structure, and sends it to the server.
5. **`void deleteUserFromContacts(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Deletes a user from the contacts list
   - **Explanation:** Takes user input for deleting a contact, constructs a `Message` structure, and sends it to the server.
6. **`void displayContacts(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Displays the list of contacts
   - **Explanation:** Requests and displays the list of contacts from the server.
7. **`void displayMessagesFromUser(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Displays messages from a specified user
   - **Explanation:** Takes user input for a specified user, constructs a `Message` structure, sends it to the server, and displays the response.
8. **`void deleteMessageFromUser(int client_socket, char *userID)`**
   - **Inputs:** `int client_socket`, `char *userID`
   - **Outputs:** Deletes a message from a specified user
   - **Explanation:** Takes user input for a specified user and message, constructs a `Message` structure, sends it to the server, and displays the response.
9. **`void clearScreen()`**
   - **Inputs:** None
   - **Outputs:** None
   - **Explanation:** Clears the console screen.