

---

# **A Secure Person2Person (P2P) Micropayment System - Client User Manual**

IM 3010 Programming Assignment: Phase 01 Implementation

Brian Li-Hsuan Chen (B07705031)

2021-12-09

## Contents

Introduction . . . . .	2
Environment . . . . .	2
macOS . . . . .	2
Ubuntu . . . . .	3
Usage . . . . .	3
Running the Client Program . . . . .	3
Registering a User . . . . .	4
Logging in to the Server . . . . .	5
Listing System Information . . . . .	7
Making Transaction with Peers . . . . .	8
Logging out from the Server . . . . .	10
How to Compile . . . . .	10
References . . . . .	10
Client-side Implementation . . . . .	10
User Manual . . . . .	11

## Introduction

In **Phase 01**, we are asked to implement a client-side function for the Person-to-Person Transaction. The existing functions for the client-side program include *registering*, *login*, *listing*, *transacting*, and *exiting*. Simply start running the program by `./client <SERVER_IP> <SERVER_PORT>` after compilation (which can be done by `make client`).

The user manual will cover the running environment used when developing the program, the environment that this code could be used in, the usage of the client-side program, the compilation, and the references when doing this assignment.

## Environment

### macOS

The environment used to develop this project is:

Operating System: macOS 12.0.1  
CPP Standard: C++11

It means that **you can run this program in a macOS environment** if the program is also compiled in the exact environment.

## Ubuntu

For the given `client` binary, you can compile and run it in:

```
Operating System: Ubuntu 20.04  
CPP Standard: C++11
```

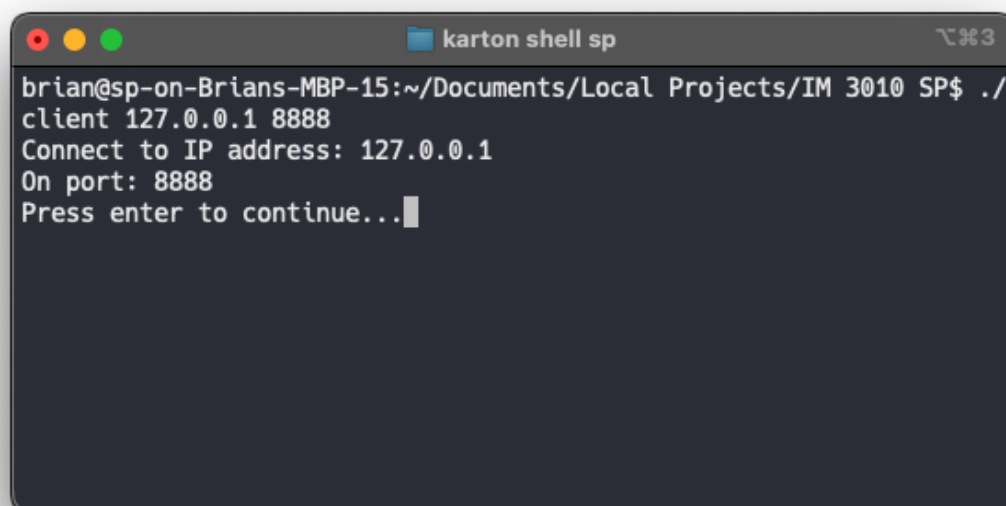
I am testing out the Linux-formatted `client` binary on Karton from my MacBook. Karton is developed based on Docker containers.

**Notice that the user interface requires Nerd Fonts to render.**

## Usage

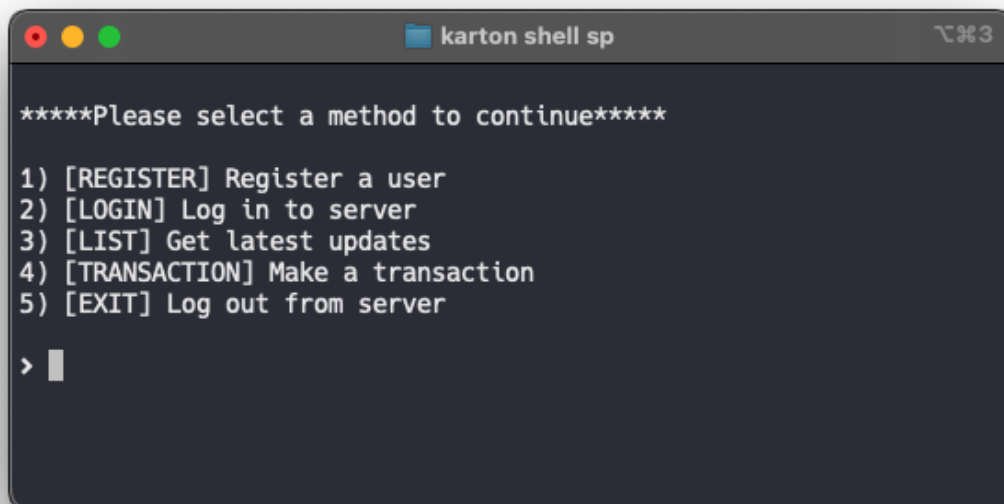
### Running the Client Program

Before running the client program, you have to make sure that the `server` is running. In the following case, I am running the client program on 127.0.0.1 with port 8888.



```
brian@sp-on-Brians-MBP-15:~/Documents/Local Projects/IM 3010 SP$ ./
client 127.0.0.1 8888
Connect to IP address: 127.0.0.1
On port: 8888
Press enter to continue...
```

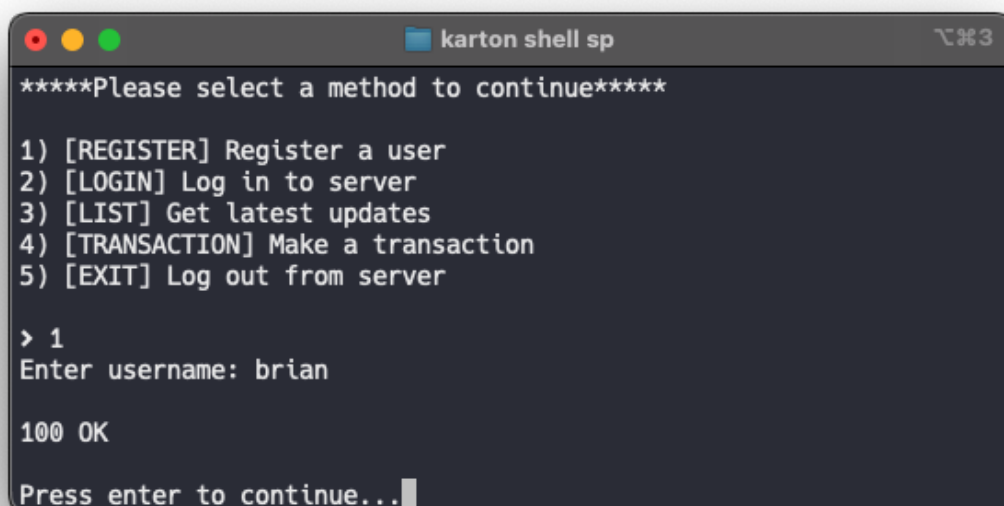
After logging in, a menu should popup:



```
*****Please select a method to continue*****  
  
1) [REGISTER] Register a user  
2) [LOGIN] Log in to server  
3) [LIST] Get latest updates  
4) [TRANSACTION] Make a transaction  
5) [EXIT] Log out from server  
  
> |
```

## Registering a User

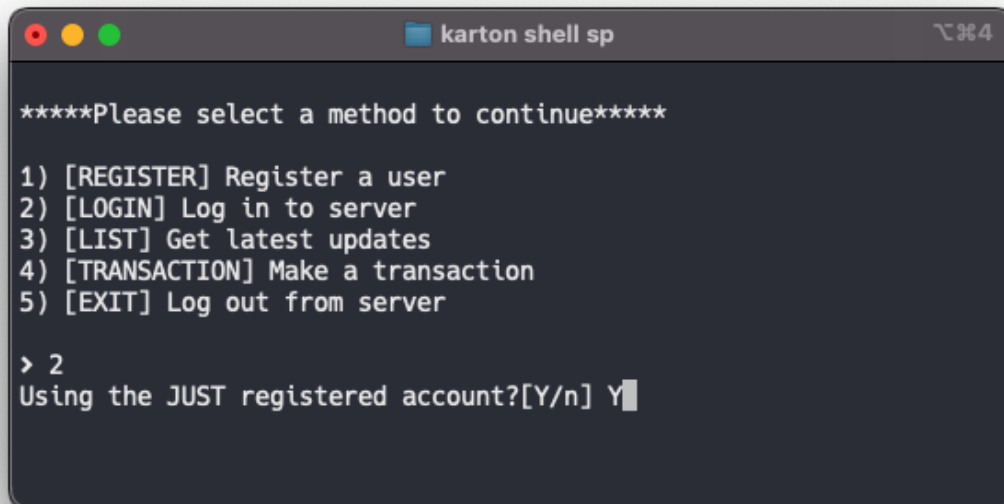
Press 1 to register a user. You can register more than one user in one go.



```
*****Please select a method to continue*****  
  
1) [REGISTER] Register a user  
2) [LOGIN] Log in to server  
3) [LIST] Get latest updates  
4) [TRANSACTION] Make a transaction  
5) [EXIT] Log out from server  
  
> 1  
Enter username: brian  
  
100 OK  
  
Press enter to continue...|
```

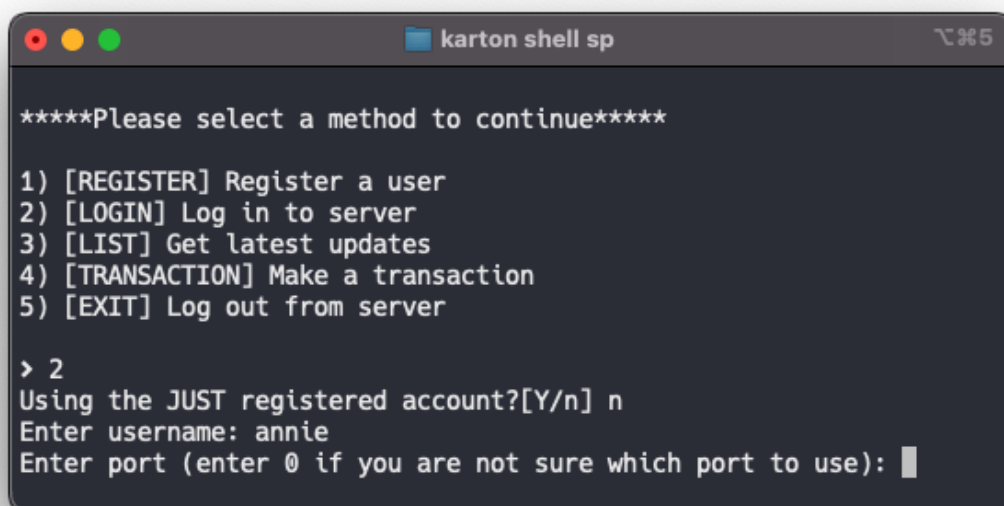
## Logging in to the Server

You can log in to a user by telling the program to use the JUST registered user account by typing `Y` (case sensitive) like what the following screenshot shows.



```
*****Please select a method to continue*****  
1) [REGISTER] Register a user  
2) [LOGIN] Log in to server  
3) [LIST] Get latest updates  
4) [TRANSACTION] Make a transaction  
5) [EXIT] Log out from server  
  
> 2  
Using the JUST registered account?[Y/n] Y
```

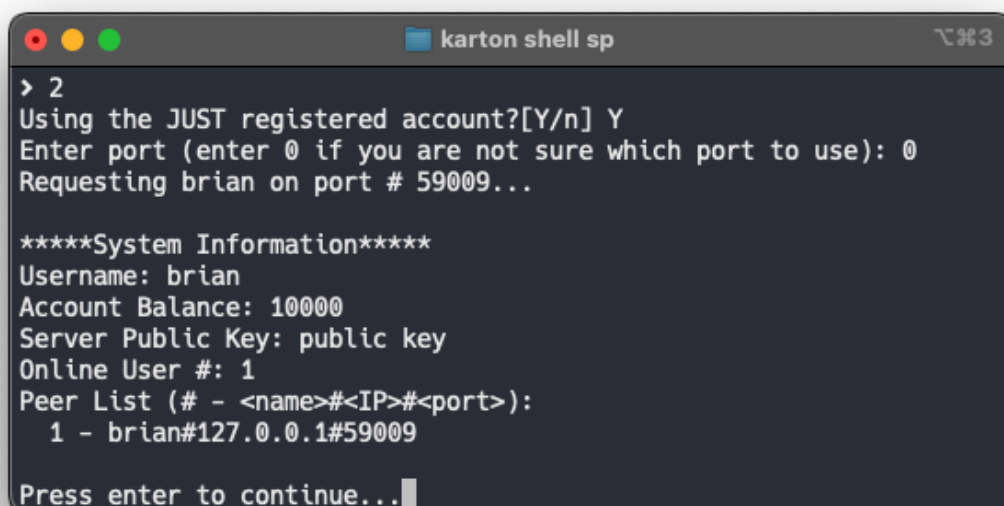
Or you can also type in `n` (case sensitive) to manually key in the username:



```
*****Please select a method to continue*****  
  
1) [REGISTER] Register a user  
2) [LOGIN] Log in to server  
3) [LIST] Get latest updates  
4) [TRANSACTION] Make a transaction  
5) [EXIT] Log out from server  
  
> 2  
Using the JUST registered account?[Y/n] n  
Enter username: annie  
Enter port (enter 0 if you are not sure which port to use):
```

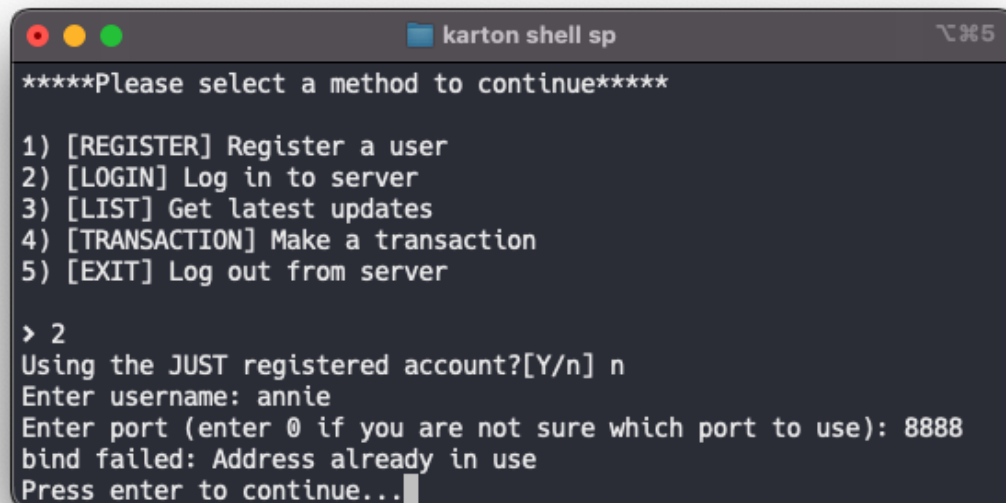
We also have to give the user a port to login on. If you enter 0, my program will assign an available port for you automatically. Otherwise, you have to make sure that you pick an available one.

The following screenshot (user [brian](#)) shows the case when 0 is recorded:



```
> 2  
Using the JUST registered account?[Y/n] Y  
Enter port (enter 0 if you are not sure which port to use): 0  
Requesting brian on port # 59009...  
  
*****System Information*****  
Username: brian  
Account Balance: 10000  
Server Public Key: public key  
Online User #: 1  
Peer List (# - <name>#<IP>#<port>):  
  1 - brian#127.0.0.1#59009  
  
Press enter to continue...
```

The following screenshot (user [annie](#)) shows the case when a custom port is entered, but not available:



```
*****Please select a method to continue*****
1) [REGISTER] Register a user
2) [LOGIN] Log in to server
3) [LIST] Get latest updates
4) [TRANSACTION] Make a transaction
5) [EXIT] Log out from server

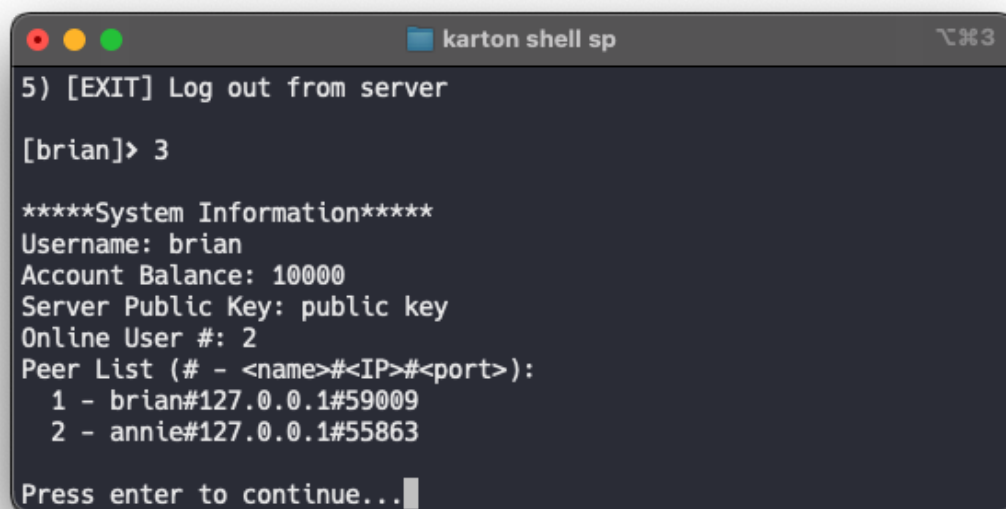
> 2
Using the JUST registered account?[Y/n] n
Enter username: annie
Enter port (enter 0 if you are not sure which port to use): 8888
bind failed: Address already in use
Press enter to continue...
```

If successfully logged in to the server, you should see that the prompt becomes [[username](#)] > rather than >.

### Listing System Information

Now that I have logged in to [brian](#) and [annie](#).

To request for a list of the system information, simply type in 3:



```
karton shell sp 3

5) [EXIT] Log out from server

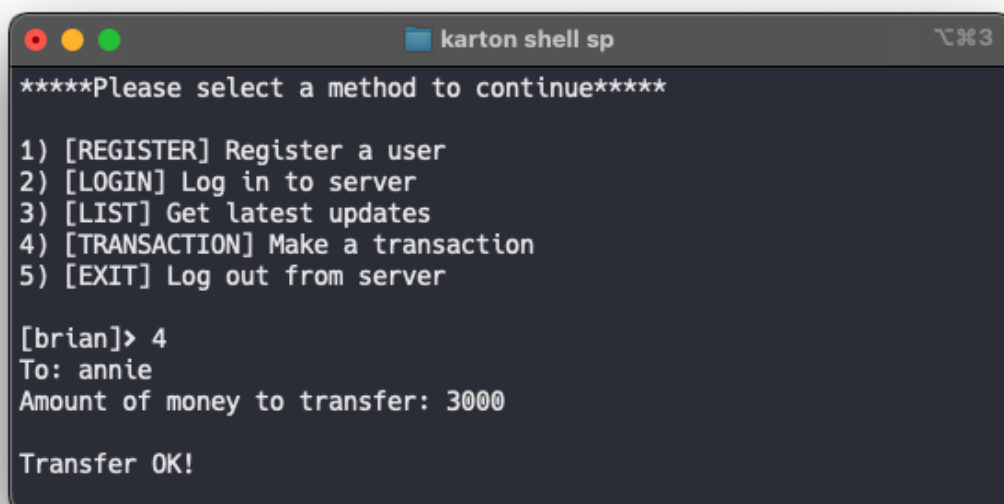
[brian]> 3

*****System Information*****
Username: brian
Account Balance: 10000
Server Public Key: public key
Online User #: 2
Peer List (# - <name>#<IP>#<port>):
  1 - brian#127.0.0.1#59009
  2 - annie#127.0.0.1#55863

Press enter to continue...
```

### Making Transaction with Peers

Making transaction with peers is simple, type in 4 to proceed.



```
karton shell sp 3

*****Please select a method to continue*****

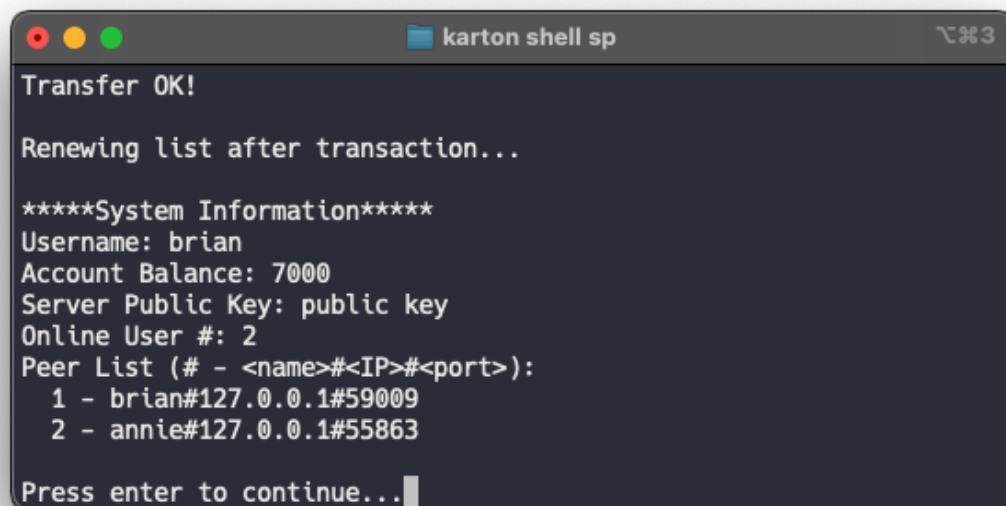
1) [REGISTER] Register a user
2) [LOGIN] Log in to server
3) [LIST] Get latest updates
4) [TRANSACTION] Make a transaction
5) [EXIT] Log out from server

[brian]> 4
To: annie
Amount of money to transfer: 3000

Transfer OK!
```



If server receives the message and acknowledges the transaction, you should see **Transfer OK!** message, and there should be another listing right after:



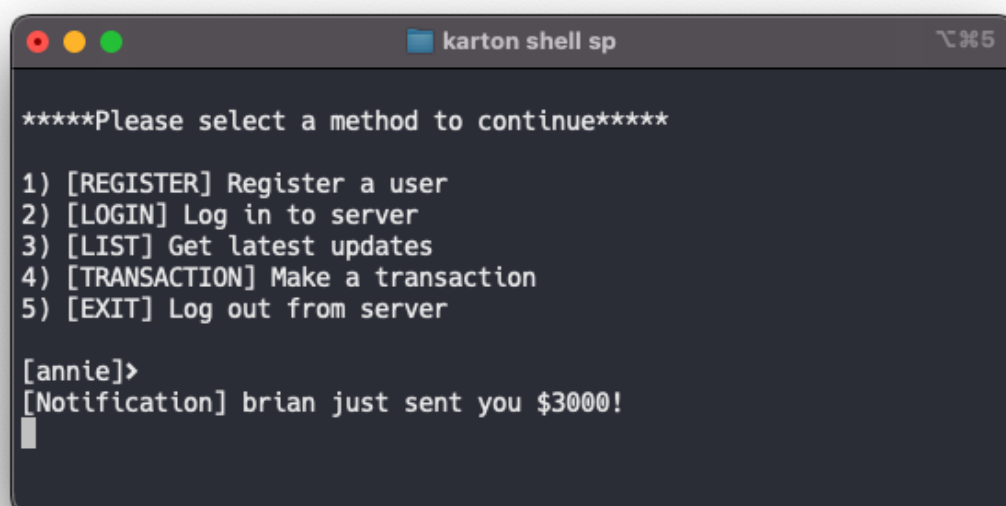
```
karton shell sp 3
Transfer OK!

Renewing list after transaction...

*****System Information*****
Username: brian
Account Balance: 7000
Server Public Key: public key
Online User #: 2
Peer List (# - <name>#<IP>#<port>):
  1 - brian#127.0.0.1#59009
  2 - annie#127.0.0.1#55863

Press enter to continue...
```

On the receiver side (the one who receives the money), there should be a notification telling that a transaction has been made.



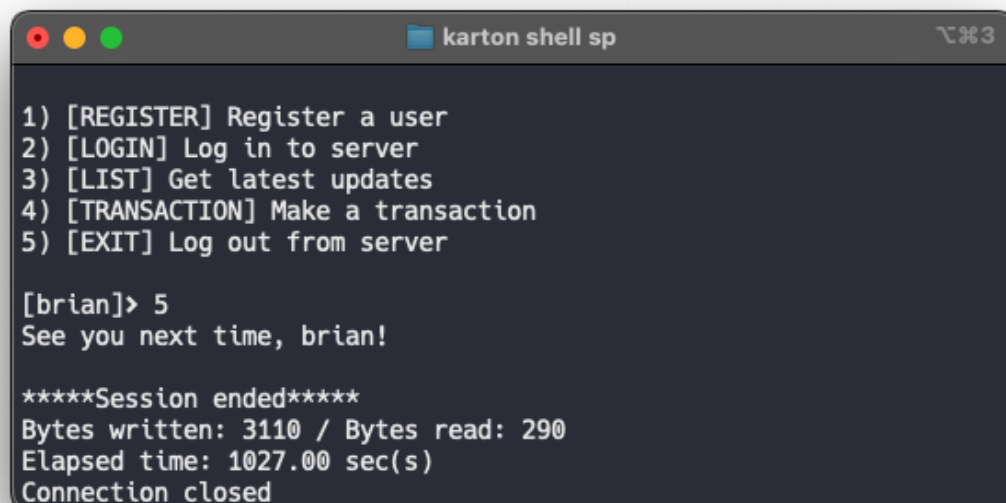
```
karton shell sp 5
*****Please select a method to continue*****

1) [REGISTER] Register a user
2) [LOGIN] Log in to server
3) [LIST] Get latest updates
4) [TRANSACTION] Make a transaction
5) [EXIT] Log out from server

[annie]>
[Notification] brian just sent you $3000!
```

## Logging out from the Server

You can log out by typing 5:



```
1) [REGISTER] Register a user
2) [LOGIN] Log in to server
3) [LIST] Get latest updates
4) [TRANSACTION] Make a transaction
5) [EXIT] Log out from server

[brian]> 5
See you next time, brian!

*****Session ended*****
Bytes written: 3110 / Bytes read: 290
Elapsed time: 1027.00 sec(s)
Connection closed
```

The program will show you a goodbye message along with *bytes written* and *read* and the *elapsed time* in this session.

## How to Compile

You can start the program already by typing `./client <SERVER_IP> <SERVER_PORT>` into your terminal if you are on a Linux distribution (*Ubuntu* is used for testing).

To rebuild the program, on either **macOS** or a **Linux** system, run `make client` in the terminal app.

If `client` binary already exists, you may want to run `make clean` first to remove the file.

## References

### Client-side Implementation

- Repositories
  - Learn Network Protocol and Programming Using C at <https://github.com/apsrcreatix/Socket-Programming-With-C>

- Peer to peer program in C at <https://github.com/um4ng-tiw/Peer-to-Peer-Socket-C>
- C Multithreaded Client-Server at <https://github.com/RedAndBlueEraser/c-multithreaded-client-server>
- Socket programming examples in C++ at <https://github.com/zappala/socket-programming-examples-c>
- Others
  - Parse (split) a string in C++ using string delimiter (standard C++) at <https://stackoverflow.com/a/14266139/10871988>
  - Finding Unused Port in C++ at <https://stackoverflow.com/a/1107242/10871988>
  - Unix Specification (link to `bind()`) at <https://pubs.opengroup.org/onlinepubs/007908799/xns/bind.html>, but of course many more functions are looked up
  - Port Forwarding for a Docker Container at <https://docs.docker.com/config/containers/container-networking/>
  - Karton for not running on virtual machine at <https://karton.github.io>

## User Manual

- Eisvogel at <https://github.com/Wandmalfarbe/pandoc-latex-template>