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

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

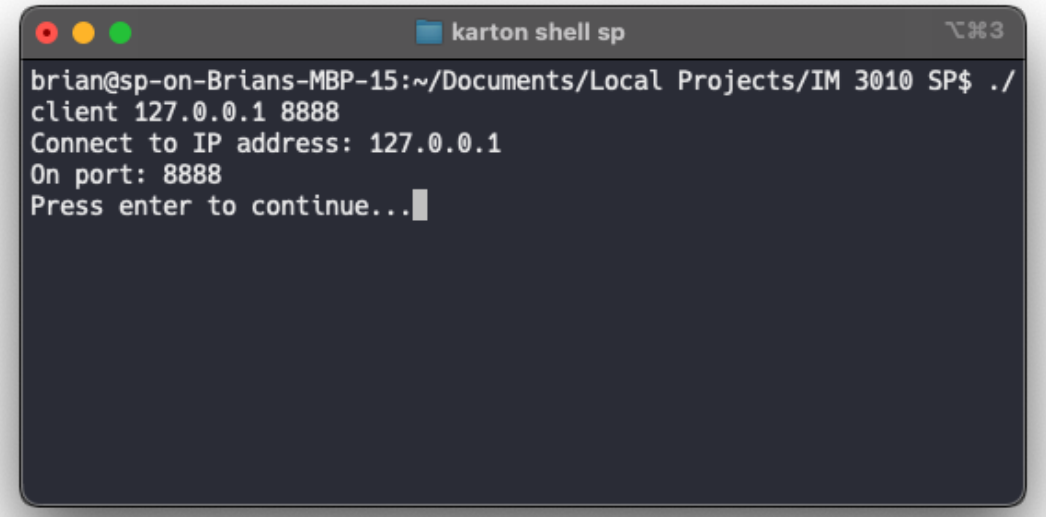
I am testing out the Linux-formatted `client` binary on Karton from my MacBook. Karton is developed based on Docker containers.

Notice that the user interface requires Nerd Fonts to render.

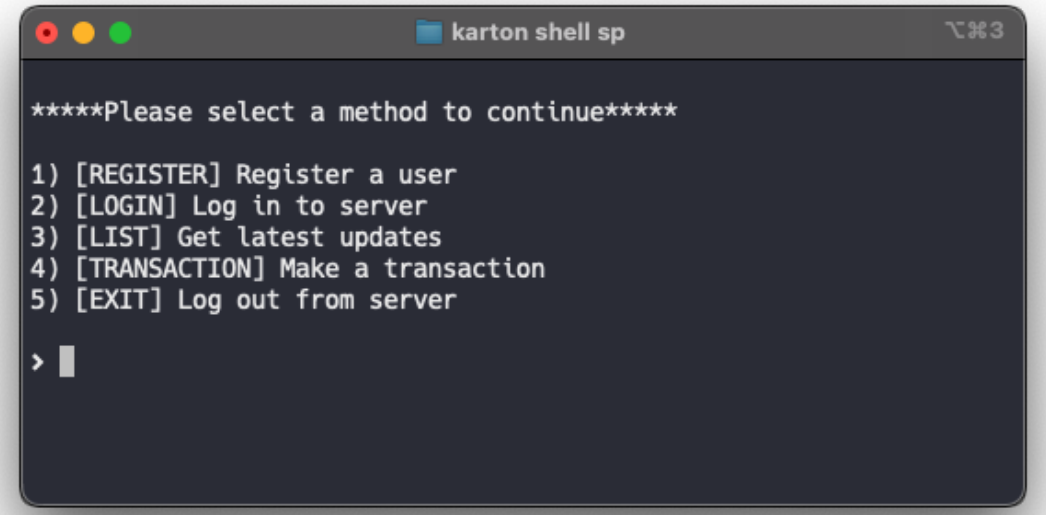
Usage

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.

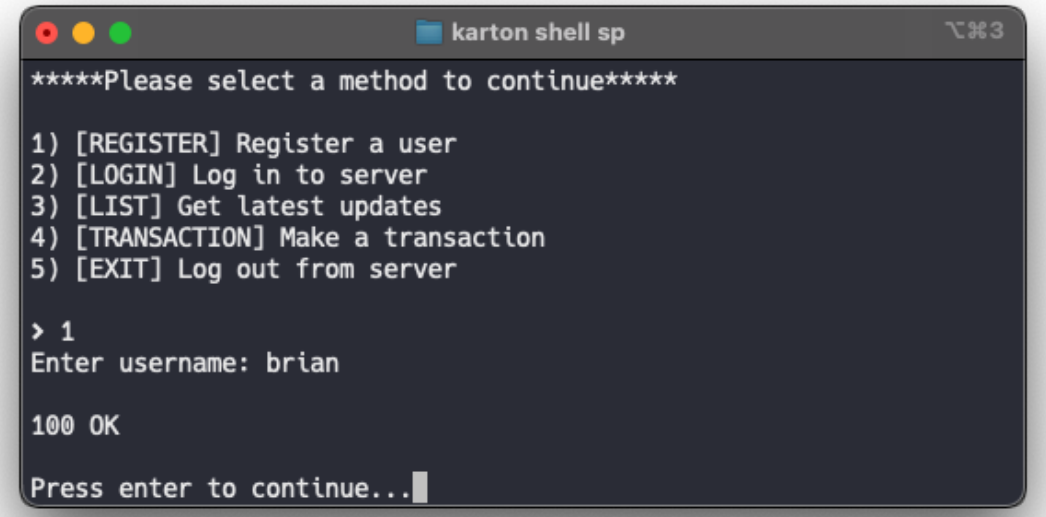


After logging in, a menu should popup:



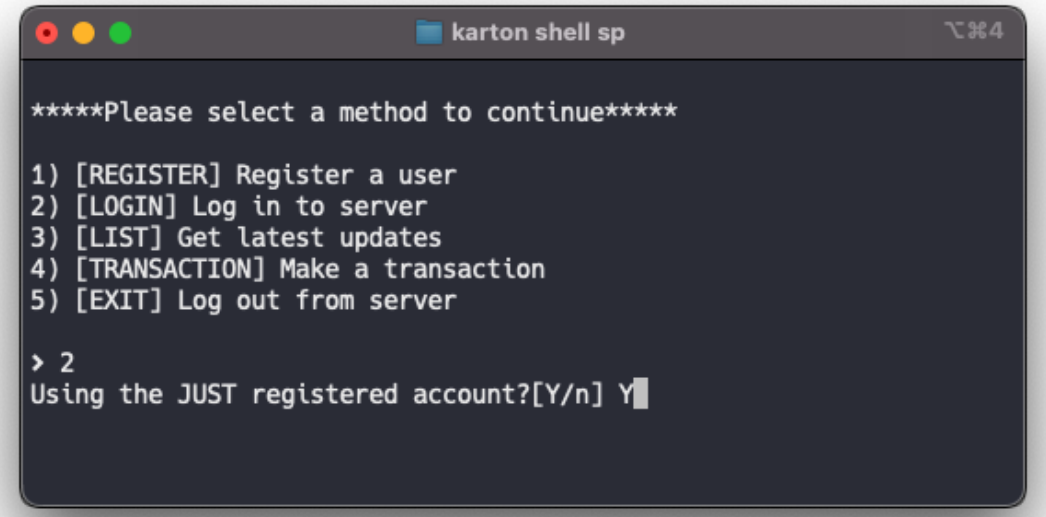
Registering a User

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

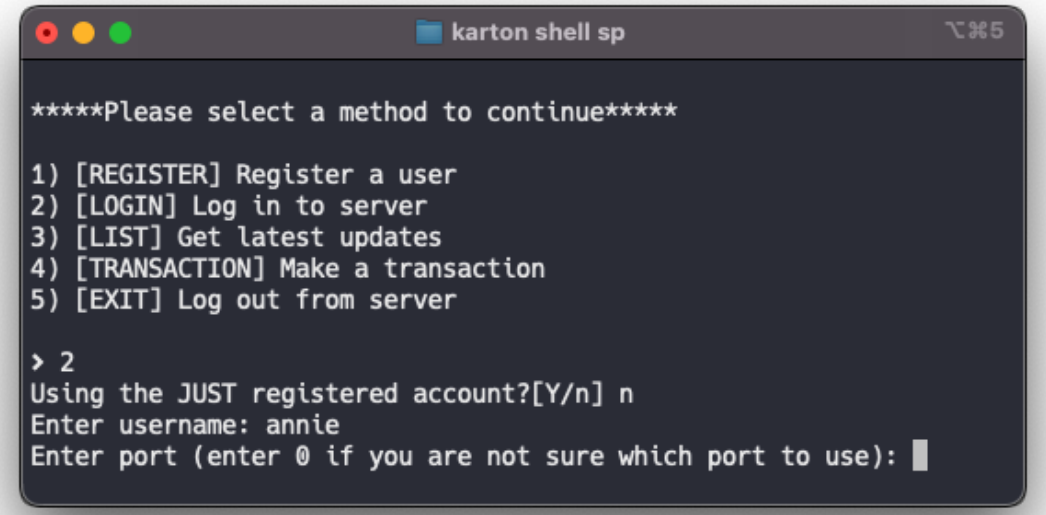


Logging in to the Server

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.

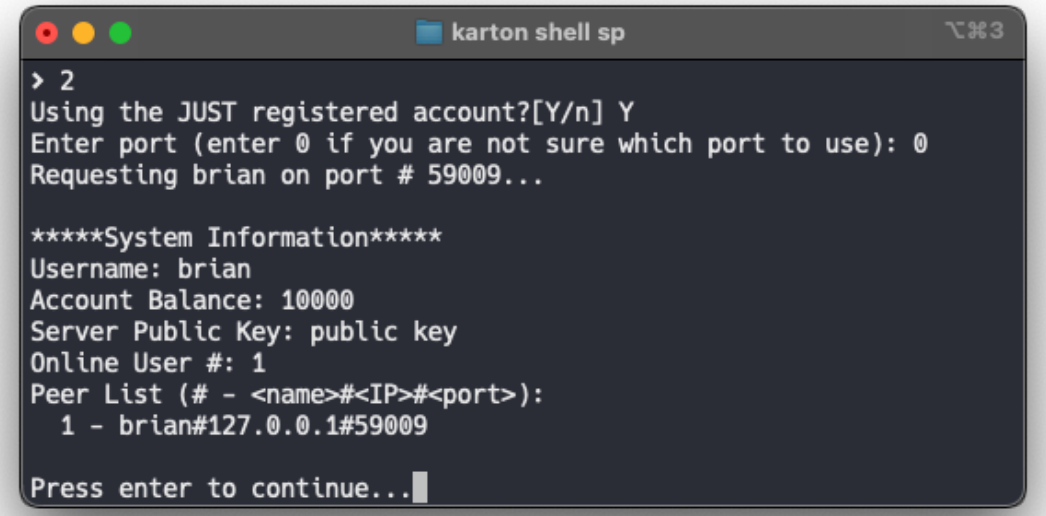


Or you can also type in `n` (case sensitive) to manually key in the username:

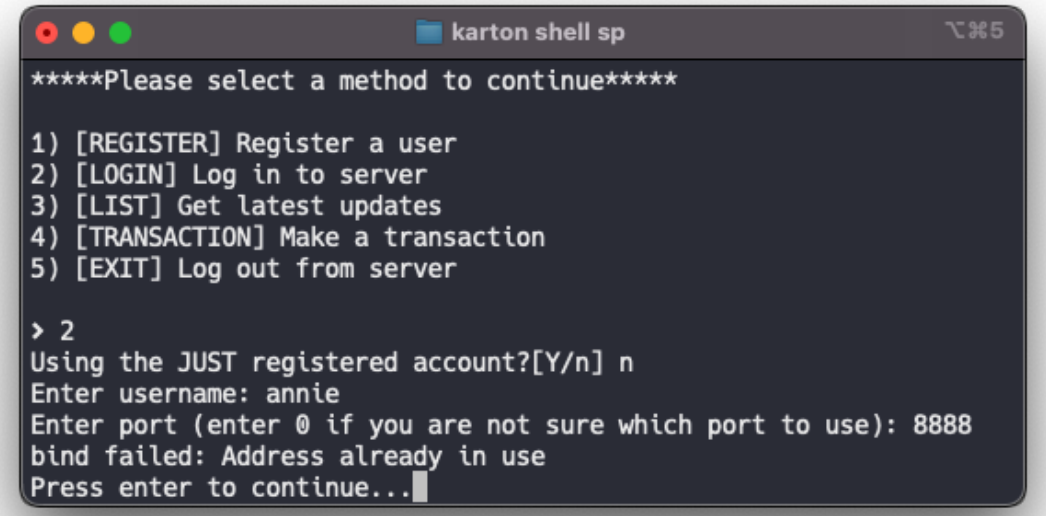


We also have to give the user a port to login on. If you enter `0`, my program will assign an available port for you automatically. Otherwise, you have to make sure that you pick an available one.

The following screenshot (user `brian`) shows the case when `0` is recorded:



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

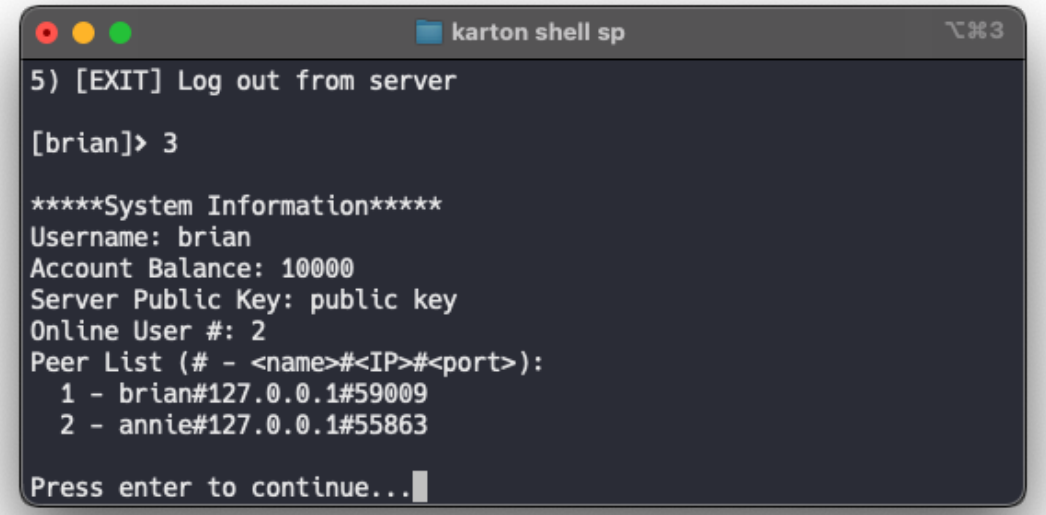


If successfully logged in to the server, you should see that the prompt becomes `[username]>` rather than `>`.

Listing System Information

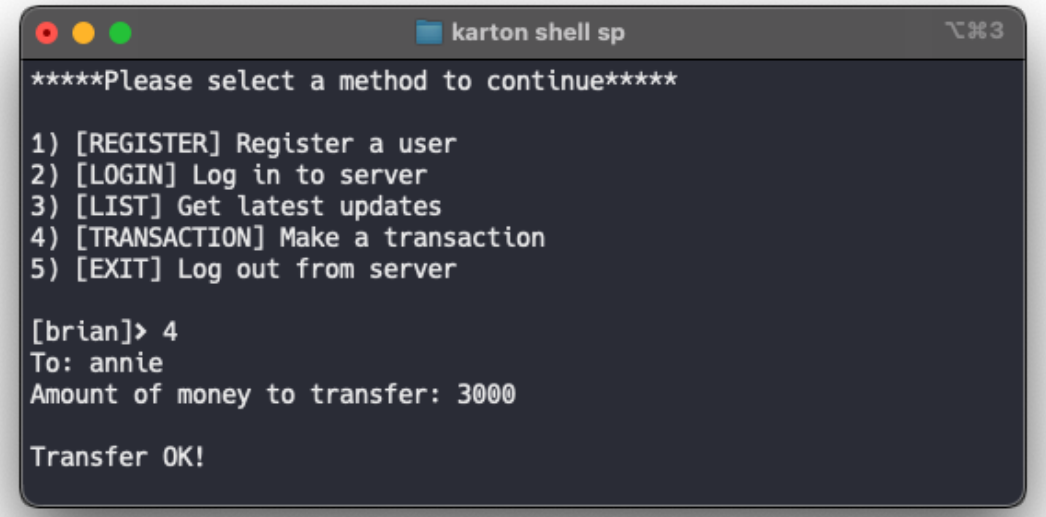
Now that I have logged in to `brian` and `annie`.

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

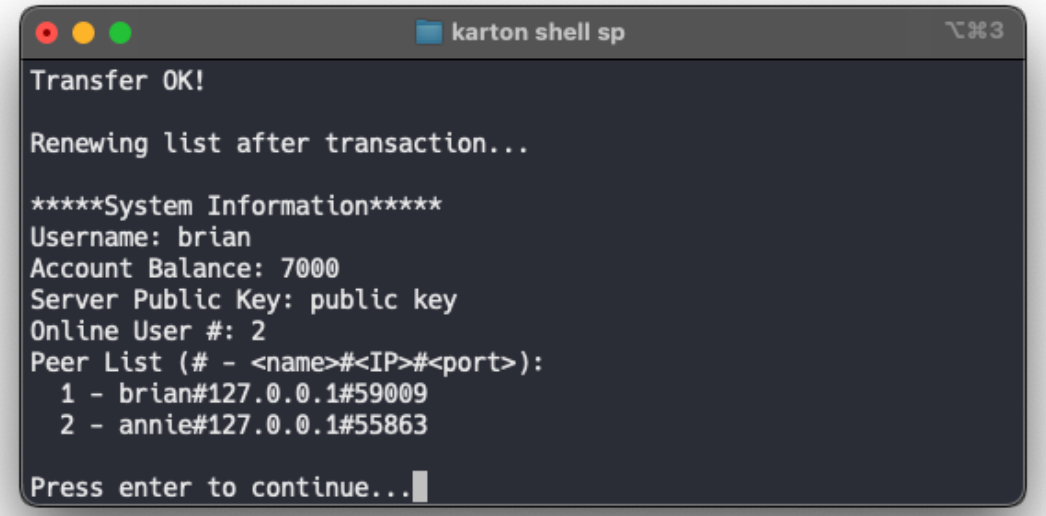


Making Transaction with Peers

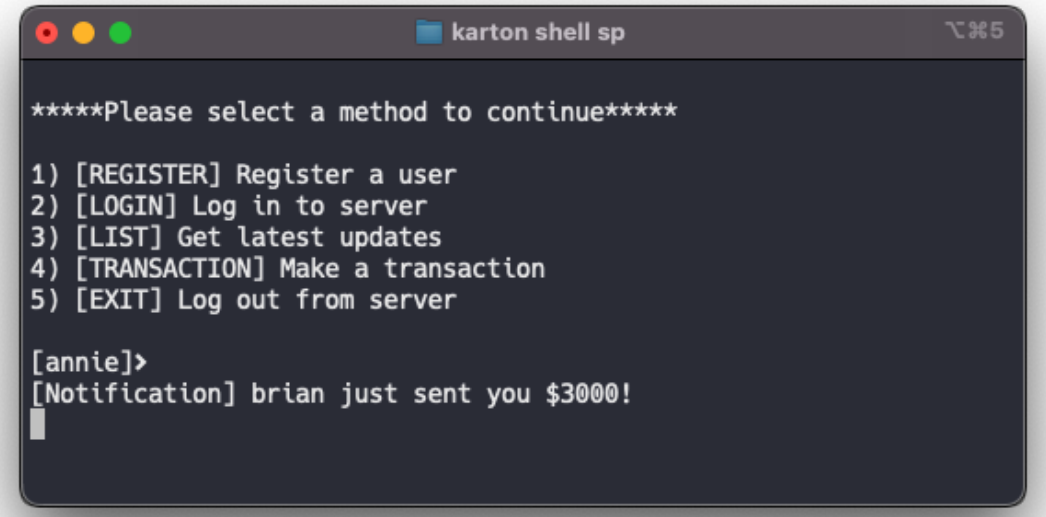
Making transaction with peers is simple, type in 4 to proceed.



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

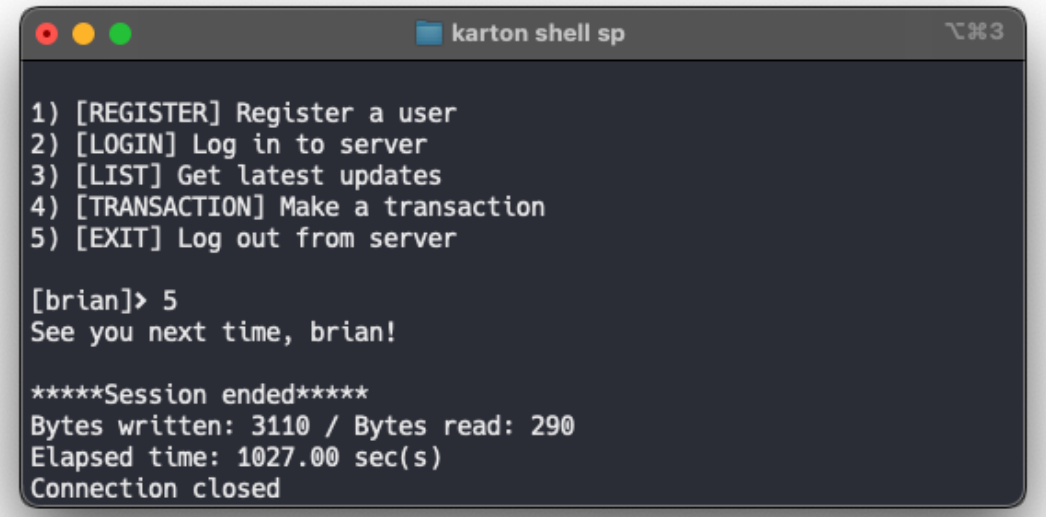


On the receiver side (the one who receives the money), there should be a notification telling that a transaction has been made.



Logging out from the Server

You can log out by typing `5`:



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