

Social Media Network

Peter Joe | COMPSYS302

# Features

## Basic Requirements

1. User can log in
   1. The user must enter their username, password, location, and an additional 2FA code to login.
2. User can see who is currently online
   1. Upon login, the user can view other users on a sidebar on the right. This sidebar is refreshed every 8 seconds due to rate limiting.
3. User can see and edit their profile page
   1. Once logged in, the user can see their profile page on a sidebar on the left. They can edit it once the edit button is pressed.
4. User can send, receive, and view messages and files with someone online
   1. Any item sent to the user is stored in a database, with this, the server will read the database and display a message depending on the type of item it is (message, photo, pdf, video, audio)

## Features that imrpoved functionality

1. Automatically refreshing page (or refreshing content) and/or notifications
   1. Using Javascript, the chat is continuously refreshing every 2 seconds, while the online user sidebar is continuously refreshing every 8 seconds. Both are done without refreshing the entire page.
2. Good use of database(s)
   1. The database contains four tables:
      1. Online users: Stores all online user details (IP, port, UPI, etc)
      2. Users: Stores the name of all users and their authentication code
      3. Message: Stores all messages
      4. User profile: Stores all users profile details (if received)
   2. With these four tables, data is stores efficiently
3. Use of encryption/hashing/data security within application
   1. Once password is gained, it is then encrypted with the username as a hash and is passed to the login server
4. Retrieve profiles for other users, provide profiles to other users
   1. Once a user clicks on another user’s profile, they can see either:
      1. Their current profile if they are online (so the server can retrieve their profile)
      2. Their profile as last recorded on the database (the user profile table contains default values if the desired user’s profile has never been retrieved)
   2. If JSON is sent to the getProfile API, the server will return the current session’s users information to the requester
5. Displays confirmation of message receipt
   1. Upon sending another user a message, the time is recorded and is then displayed along with the message
6. Error Logging
   1. With try/except, if an error occurs, the error is logged into a text file which is stored locally called “errorLog”
7. Rate Limiting
   1. Login server is called three times during login to enable the 2FA, and is called 7 times a minute to reupdate online users, therefore this will be under the limit of 10 per minute
8. Use of threading for communicating with login server regularly
   1. Threading is begun once the user has entered (successfully) their authentication code and stops when the user has signed out.
   2. The thread will constantly sign in the user every minute
9. Embedded viewer for image, audio, video, and PDFs
   1. With each of these, in the chat there is a way to view each as they each have their own html tags assigned to them depending on their content type.
10. (Good) Page Templating
    1. Throughout the website, there is a consistent design and layout of each page.
11. Modular and Pythonic code, including commenting and documentation
    1. While some sections may not look neat and tidy, they are pythonic code as the style written out is more like python than C++.
    2. Each function has a comment and there are sections which describe what each section is responsible for
12. Nice user interface (preferably compatible cross-browser)
    1. The interface follows a color scheme of orange and black, with everything condensed into one page
    2. Regarding cross-browser, almost everything works, except some files which do not appear (i.e. .wav or .ogg in IE)
13. 2FA (Two Factor Authentication)
    1. After logging in, the server will send an email to the student email which contains a code which the user must enter.
    2. This 2FA works under the assumption that everyone reported in the login server’s getList has an email associated with Auckland University and that their login is included in said email.
14. Fails graciously when interacting with substandard clients
    1. When an error has occurred, a cherrypy redirect occurs.
    2. If the user calls a function within the server with incorrect parameters, a 404 message will appear.
15. Defense against injection attacks
    1. SQL Injections:
       1. Within every database execution, the question mark notation is used to prevent input commands from being recognized and executed.
    2. HTML/Javascript Injections:
       1. With every input that will be displayed, the character “<” is replaced in order to prevent the formation of creating a HTML tag or a Javascript script.

# Top-level view



# Issues

* Threading was an issue because it was a difficult idea to grasp, while creating and running a thread was simple enough, looping it and being able to stop a thread at any given point was a trickier concept. This however was overcome by constant research on the topic, and numerous amounts of testing.
* Refreshing content was also an issue. Obviously, the user wouldn’t want to refresh the entire page to see how many users came online therefore there needed to be a fix. Although it was suggested that AJAX was used, XMLHTTPrequest was used and it was vanilla javascript. This was solved through many trials and attempts at exporting the data as JSON so XMLHTTPrequest could read it.

# Peer to Peer method

Regarding the given peer to peer methods, the only compulsory method that maybe could be removed is /ping, only because the way ping is used in this system does not incorporate anything into ping, ping does not update when a user was last seen or limit requests, while other systems may be different, there could just as easily be a function that checks whether the user is still online when refreshing the online users. Regarding other functions, they improve the functionality of the system, and overall, they seem to be well suited for a messenger type application. In addition, if this system was based solely based on peer to peer, security would become much tighter, where the middleman is removed (the login server).

# Protocol

The protocol involves using both a login server and peer to peer. As mentioned earlier peer to peer is safer in terms of security as information is only being passed through two users instead of an additional third-party group (the login server) which could leak information. In terms of the actual protocol, it is essential that a protocol be made between parties so that the get or post methods all operate the same so that each user is compatible with all others.

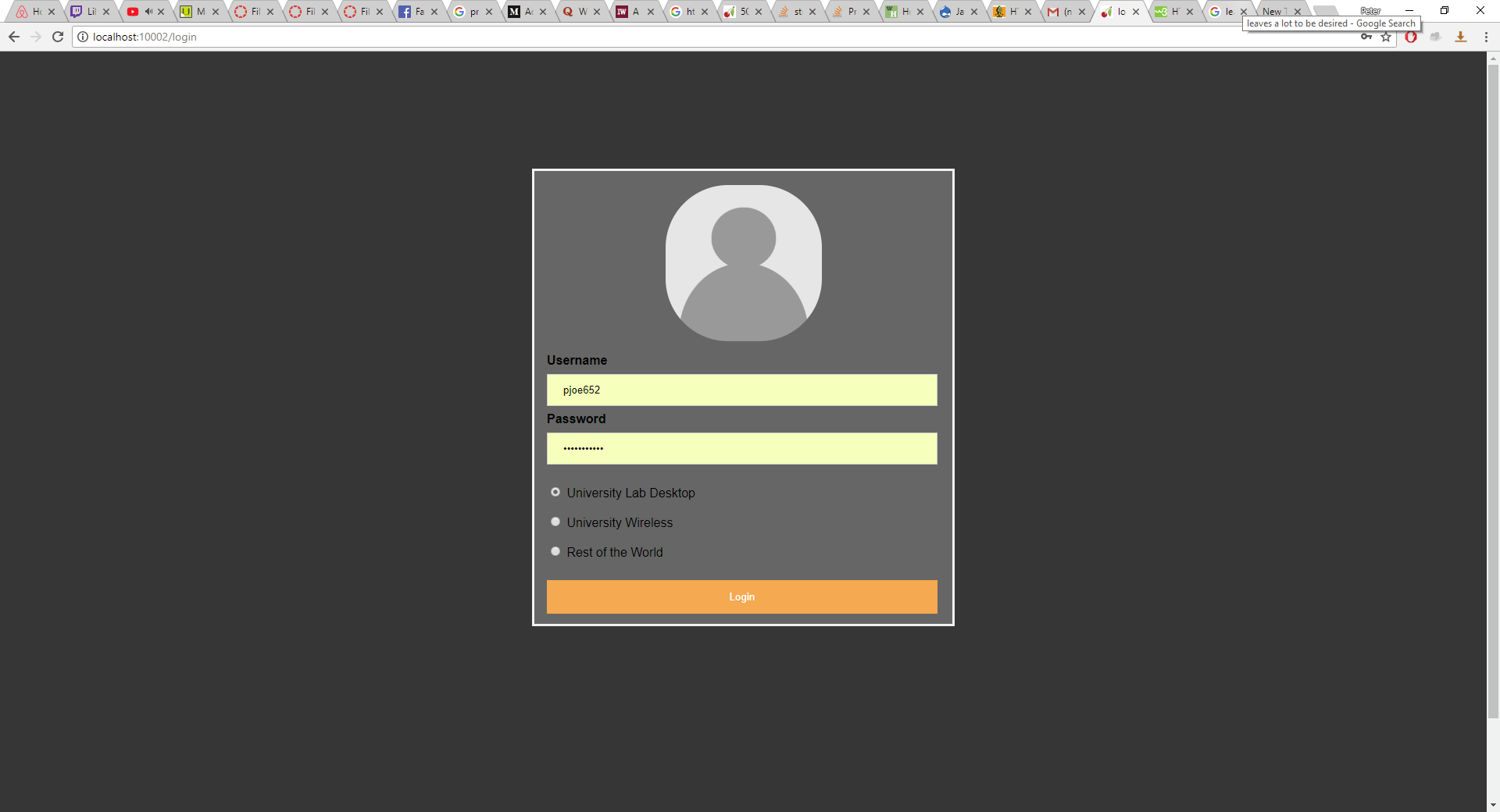
# Suitability of tools

Python is developer friendly, in terms of having simple syntax, readability, and flexible. Due to these reasons, it becomes easy to transition to an object-oriented web framework such as CherryPy, where, to make this application work as efficiently as possible, we would need to have a good understanding of Python. There are also the integration features, Python also integrates Enterprise Application Integration making it easy to develop Web services and can process markup languages which is used extensively in this system.

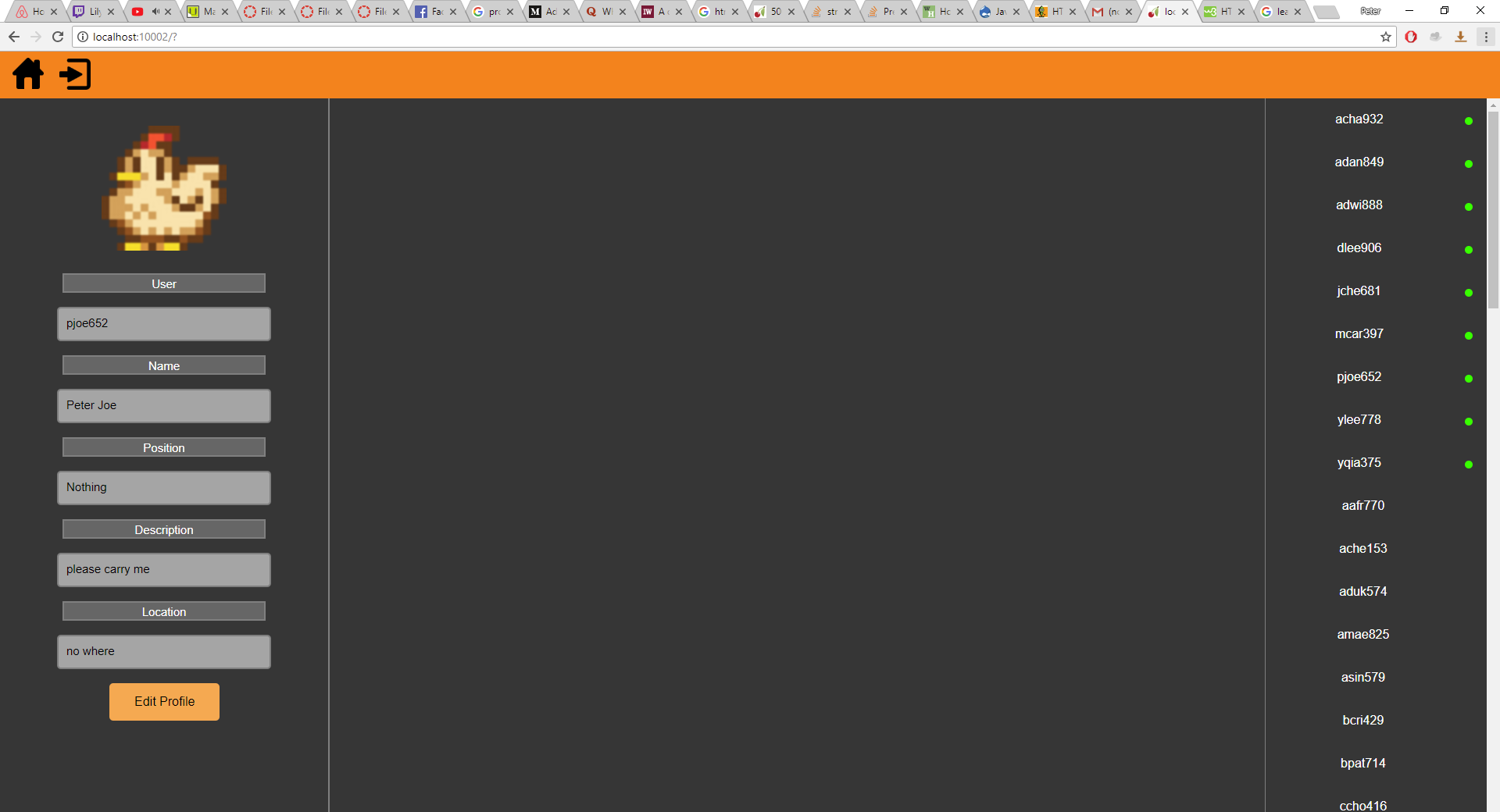
# Suggested improvements

Of course, many more features can be implemented if given additional time, as of now the features that would impact the application significantly are a fallback P2P network, high standard encryption, offline messaging. Regarding improvements, if possible, remove the rate limiting so that bar that displays online users is more accurate, make the application on all platforms, and improve the standard of code.

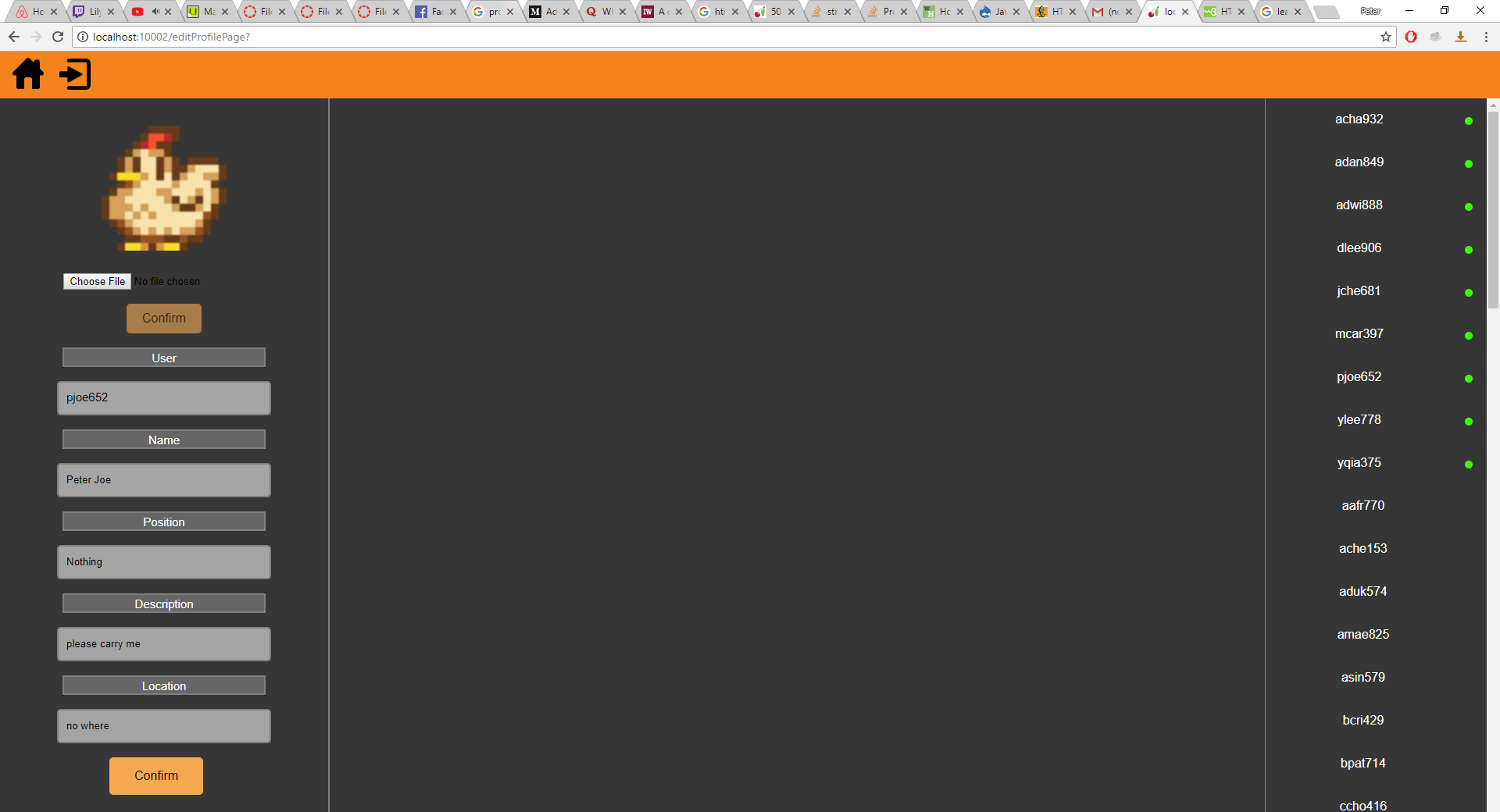
# Appendices



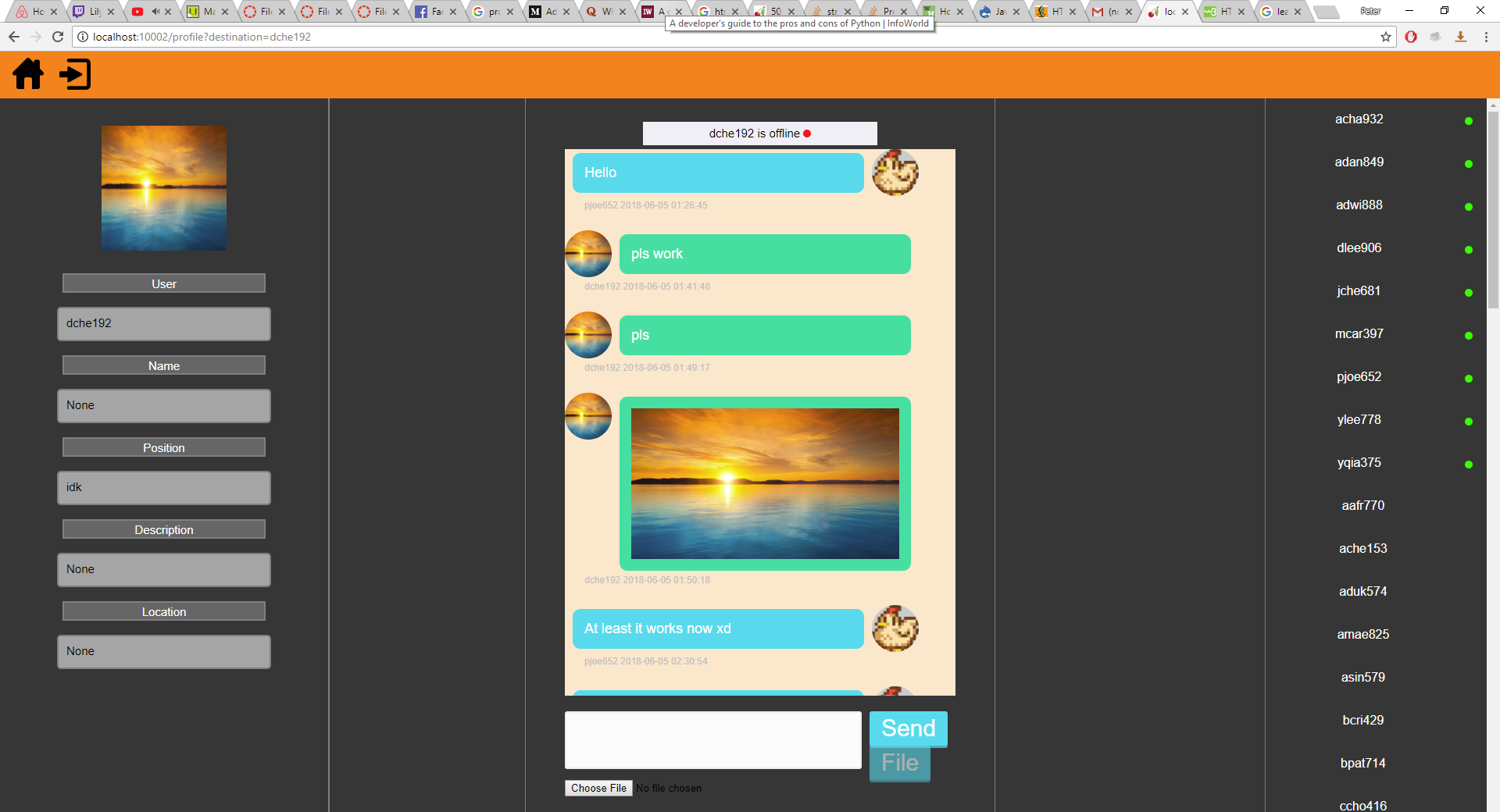
Appendix A: Login



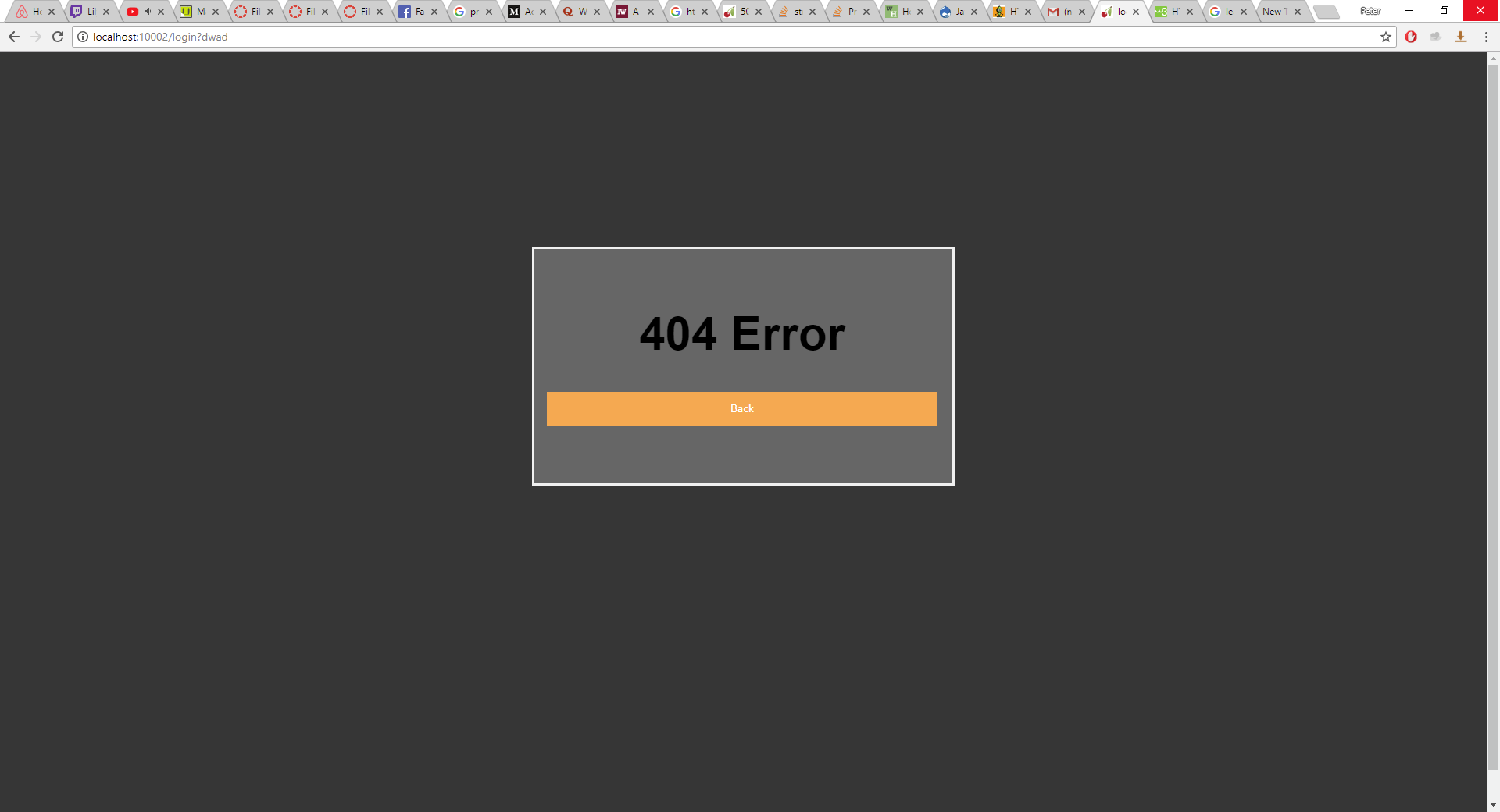
Appendix B: Home page



Appendix C: Edit profile



Appendix D: Messenger and Profile



Appendix E: 404 Error page