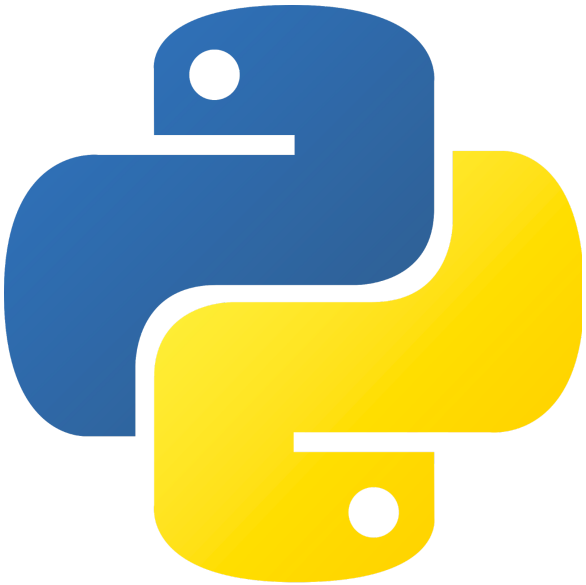

COMPSYS 302: PROJECT 2

Peer to peer social media platform

bcri429

Introduction

The goal of this project was to make a prototype peer-to-peer social media platform. The platform is intended for use in a business context by an administrative team. The aim was to provide a prototype which can help to mitigate corporate espionage within the client's company.

Requirements Met

The prototype provides a reasonable base system to protect the clients data. By nature of a peer-to-peer network only those with prearranged credentials can interact with the system. This means it is inherently safer than most other models using more central server based systems as it doesn't need to route data through a third party before they recipient gets it.

As for the application itself, it meets the functionality requirements too. Users can login to the server by use of their credentials provided to them by the admin. Once logged in they can see and edit their own profile. There is also a user list on the left of the screen which the user can interact with to receive each user's profile page. While on another users page, a chat log will appear which allows users to send and receive messages and files with said user. When the user is finished with the application they may log out from the server

Improved Features

The application has a range of extra features to increase the functionality of the prototype.

On the back end, the prototype has an extensive database the user can interact with, allowing for a full history of data received from interactions with other users. It also has error logging, meaning if something does go wrong it can be easier to diagnose. The system also uses page templating making these pages easier to reuse in the future. There is also some defense against HTML, CSS, JavaScript and SQL injections.

As for on the application side, users can also receive other users profiles, rather than just their own. The messaging system has built in message receipts so that they know if the other end has received their messages.

Peer-to-Peer Method

Using a peer-to-peer method for our system is a logical choice since exterior interference with the users data is the main concern. More specifically our server is of the ‘login server’ type in that it requires users to login through sending their credentials to a third party but nothing else.

We can check the credentials of people who interact with our server against the login server and if they don’t belong we simply need not interact with them. With this method we can rest assured that as long as we can trust someone is who they say they are, our data is much more secure. The peer-to-peer messaging also means our messages and files are not routed through another party. This could be compared to a pure central server system where anything we send can be read by the server we route it through. If this server becomes compromised then everyone's data is, whereas within our system breaches would only affect the users compromised.

Top Level View

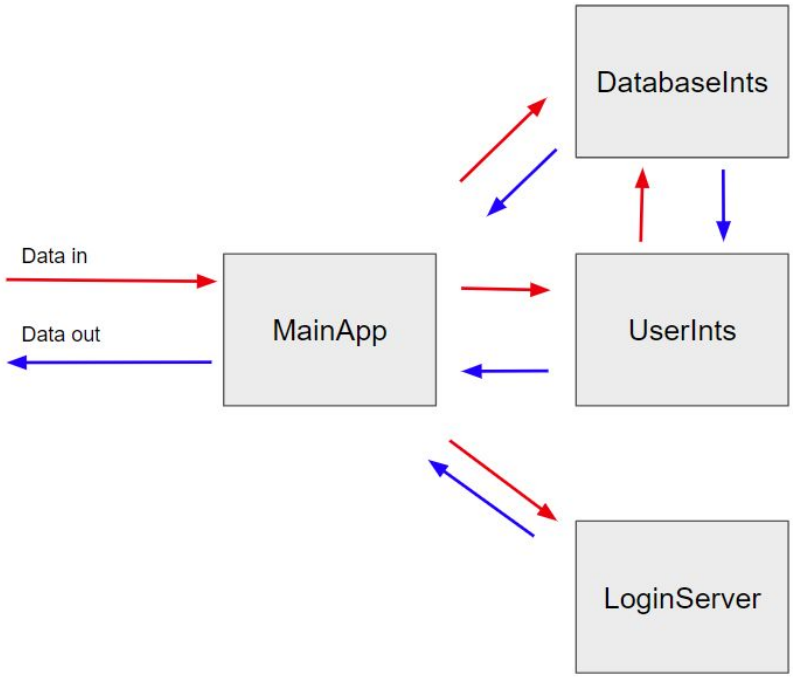

Figure 1: Top level view of system.

The system and cherrypy server is run through the MainApp class shown in figure one. This routes all incoming data to their respective areas. This also handles the templating of what we want to show to users.

When the user first tries to log in, their credentials are sent to the LoginServerClass. If their credentials are correct they will receive confirmation and be routed into the application. This login server will then provide them with a full list of online users. When the user wants to log out, this class will also send that request.

When the server receives data from other users it sends it to the databaseInts for storage, in the case of a POST, or grabs data from the databaseInts, in the case of a GET. This database has functionality to store messages as well as user profiles.

When the user themself interacts with the application these commands are sent to the UserInts class. This class handles the outgoing requests to other users, whether it be pinging, messaging, sending files or getting profiles.

Significant Issues

One of the major issues encountered was struggling to combine a large technology stack. This is the first time I have integrated multiple different languages to form a single application. I found a lot of the basic tutorials required at least a small amount of prior knowledge in one or more of the languages we using. This made it difficult for as I was relatively new to all of these languages. Even simple things like calling a cherrypy function from HTML was confusing at first. I think part of my problem was I was not used to reading the documentation for external libraries.

Another area I struggled with was time management. I initially thought that a large amount of my work was done earlier than it truly was. When I came to testing and adding together all the pieces, I found there were far more flaws than I had earlier realised meaning I would have to spend more time fixing things than I had allocated.

Protocol and Suitability

The protocol for this system was defined by the entire class. It handles all the required interactions between the user and the server as well as between users themselves. It has an

extensive range of functions well beyond what is required from the minimum specification of the system. This allows for expansion to make the prototype more robust. There is a particular emphasis on security within the protocol which is a major concern for the client. For these reasons I can say that the protocol is more than enough for prototyping though it would need expansion to fully realise the intended system.

Protocol Development

The protocol was designed by meeting in small groups and discussing the requirements of the client. From this we created protocol proposal which were sent to our manager to collate into one full document.

This allowed for us to understand API's better before starting the project, giving us a head start for when we actually underwent the project. As we were involved in the creation of the protocol we got to have our say in how we would like to implement features and security. As we are the developers of the system it is important we have our say in these areas so I believe this was a suitable process for protocol development.

Suitability of Tools

The bulk of the project was completed using Python. This high level language made it easy to delve into new, difficult concepts such as networking, web development and security without the hassle of low level development and debugging. The cherrypy library in Python also handled the Server request, meaning it was easier to spend time on the important functions of our prototype.

HTML and CSS were used for displaying our application. These markup and styling languages are the standard for modern web pages and as such are very suitable for the task.

SQL was used to handle our database. With the use of Python's SQLite library database interactions became trivial, meaning storing data was much easier.

Future Development

For future development I would mainly like to add more security features such as security within the local application, by encrypting stored data as well as encryption with user interactions. I would also like to design a nicer user interface.