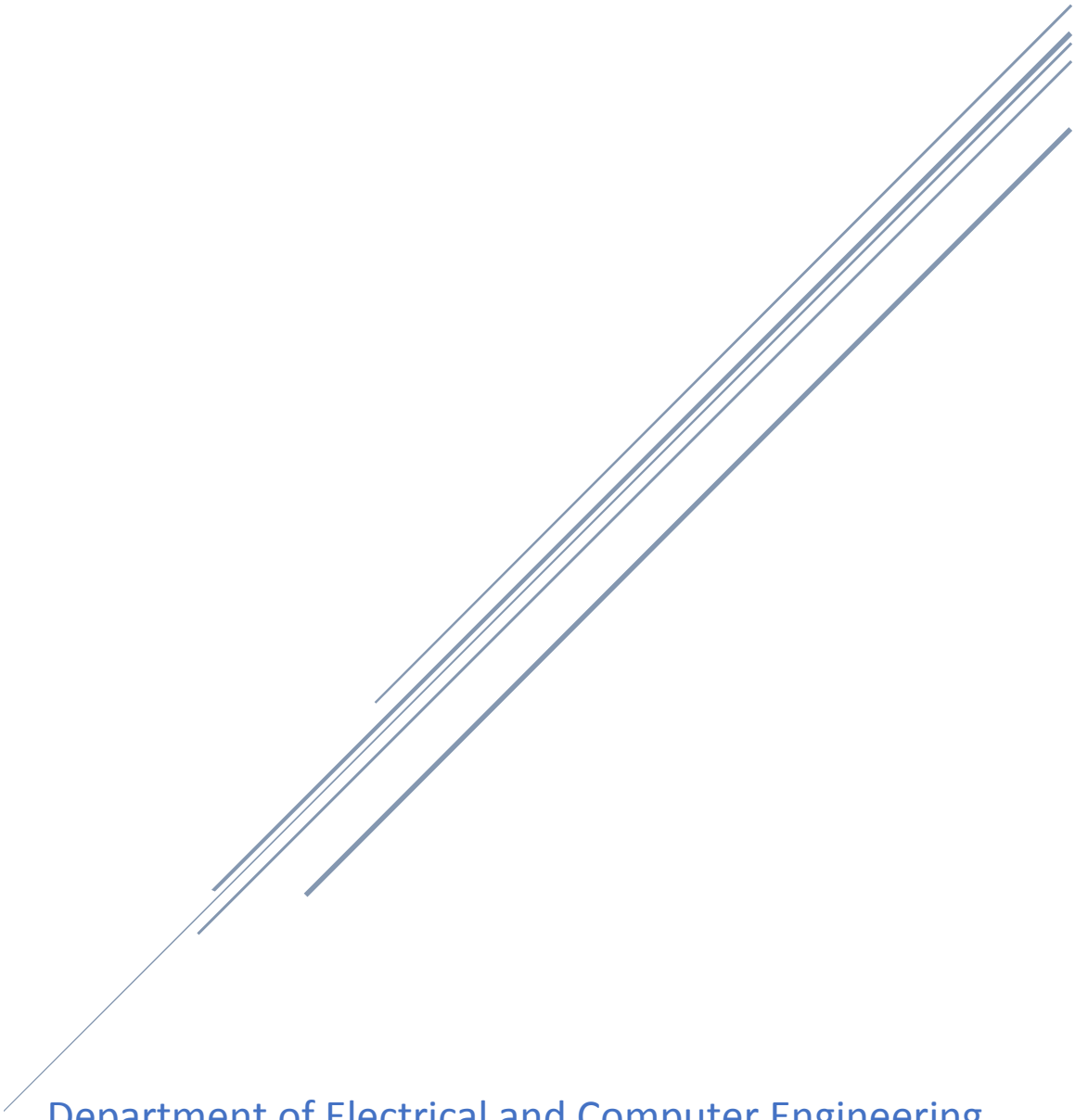
COMPSYS 302: PROJECT 2

Python Web Server



Department of Electrical and Computer Engineering
Lincoln Choy

Contents

Project Overview

How were the requirements met?

The client asked for a system which involved a social media network for his company to use without fear of being monitored. The developed system fulfils all the minimum requirements plus additional features.

The system allows the user to log in to the server and see anyone who is online. They may also view and edit their own profile and view other people's profiles. They may also send a variety of supported file formats and messages to other people using the same login server. The server also provides embedded html viewers for certain file formats, so that the clients may view the files without having to download the file and viewing it separately from the browser.

Features that improved the functionality of the system

Many of the API's and functions were grouped according to their purpose and implemented in a different file. The organisation of the files allowed for much cleaner and tidier code.

The separation of the system into different sub-modules also increased modularity of the system. Should a new system be proposed, reuse of the old components is still possible.

Significant issues during development and how they were overcome

One of the major issues encountered while developing this system was in the front-end side of the development. I wanted to try to only refresh the chat box without refreshing the entire page, every time a new message was received and when a message was sent. I learned that I had to use JavaScript as a tool to help me implement this feature: "Refreshing content". This took up a significant amount of time, and as such, I needed to compensate by removing one of the features I was planning to implement to make sure other features on my priority list would be completed by the deadline.

Another issue encountered during the development of this system was coming up with a rate limiting algorithm. As there are many ways that other nodes can call my server's API's, I had to make sure not to limit the API calls too much. To look for a reasonable number limit, I tested with a classmate by having a conversation between my application and his, I found that a rate limit of about 60 API calls per minute per person was reasonable and this was the number that I used in my application.

Technical Overview

Peer to peer methods and its suitability for this application

While a pure peer to peer network model was proposed by the manager for this project, it was decided by most of the class that a hybrid network model (login server – peer to peer) would be much more appropriate for the purposes of this project.

Having a login server makes the login part of the system much easier to implement as it was implemented by one person and therefore everyone could use it the same way. Peer to peer communication, when implemented properly, is safer due to the absence of a central server. While a pure peer to peer network model could have been used, many agree that its implementation may not have been as feasible due to developer time constraints.

Current protocol and its suitability for the system

Currently there are no problems to be seen with the login server protocol, as it mostly relates to the login server features and is only occasionally used to create group chat ID's. Most of the information given by the login server is used within the client application.

The application protocol was suitable for the purposes of this system, as the final protocol document had minimal ambiguity and most developers could implement most API's which would work between applications without issue. Many arguments were left as optional, so that most developers would be able to work on extra features if they wanted to, without compromising the weight of this project.

Process of developing the protocol and its suitability for the system

The process of developing the protocol as a class, then as smaller groups, then having it checked by the manager was relatively suitable for this project. Having a discussion in smaller groups helped increase generation of ideas, since a discussion in one large group may have had more contradicting ideas and less coherency. It was a much better idea to have separate groups propose one group protocol and submit it to the manager for checking, so that the manager would be able to compile all the good suggestions into one finalised protocol document.

The downside to this method of developing the protocol is that people from different groups may have been able to improve upon an idea that another group came up with, if they were in the same group.

One way that this process may have been improved is by having a final class discussion on the compiled protocol document and considering suggested improvements to the protocol.

Suitability of the suggested tools for the application

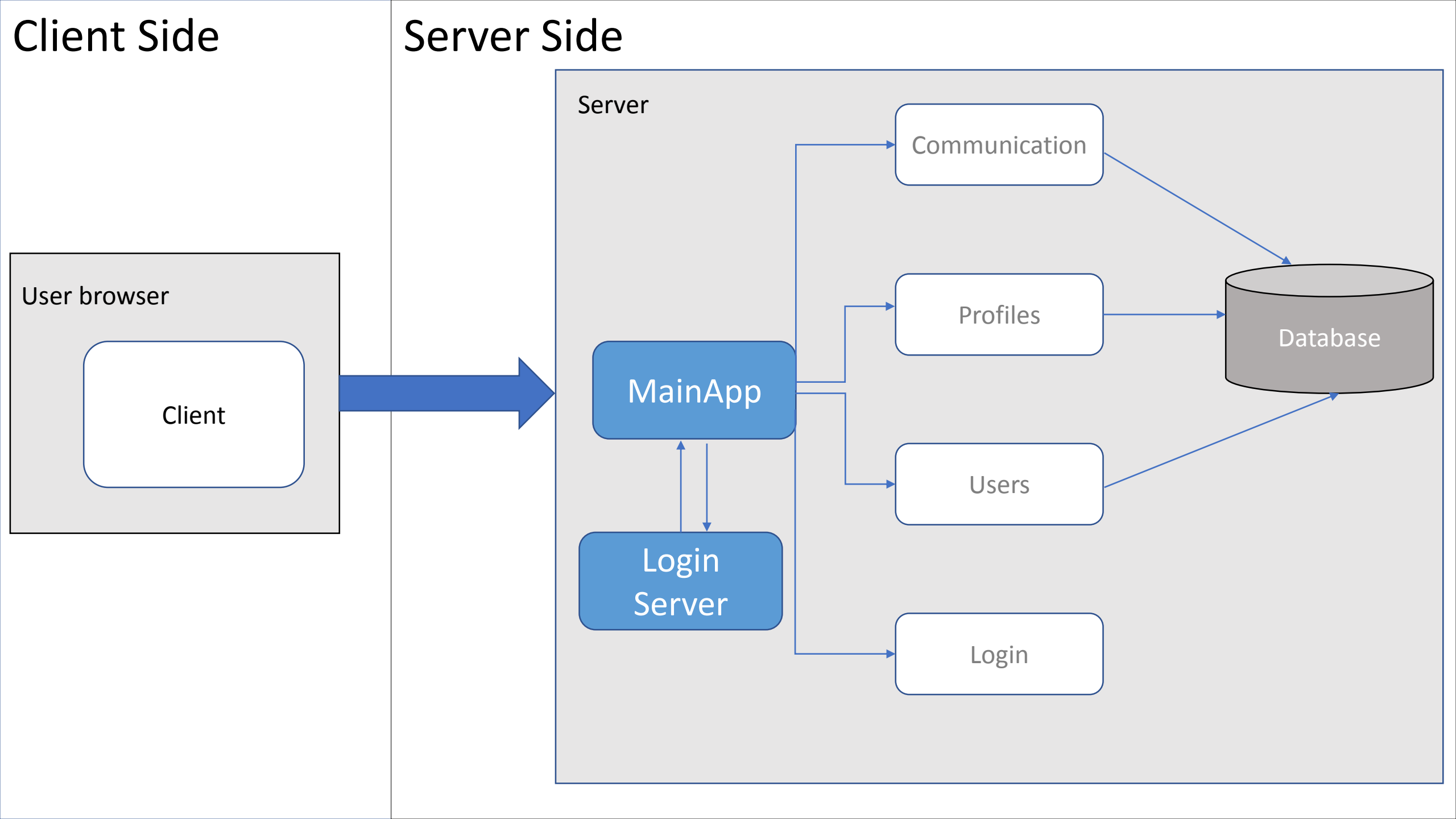
Python was the programming language used for this application. It was a very suitable language for this project and as a higher-level programming language it offers many tools for developing a server-side web application. It has frameworks which are very good for building web applications, such as Django and Cherrypy, the latter being used in this project. Also, Python's standard library supports many internet protocols, such as HTML, XML and JSON.

JSON was used to transmit data objects instead of XML, and this was much more suitable for the application as XML is just too outdated, and JSON is much nicer to work with for most developers.

Suggested improvements for future development

Due to developer time constraints, not all the extra features on the checklist were implemented. And most unfortunately, the system lacks too much data security. The system has minimum data security as it has been proven to be vulnerable against injections attacks, and passwords are the only thing that is hashed in the application. The system does not currently support any encryption standard. The system also lacks a lot of the additional proposed inter-app features, meaning the system was more closed in terms of its functionality.

As the client asked for a system with the purpose of more security, I realise that I neglected one of the more focal points of the project. In the future, I plan to continue this project with some other classmates, with a pure peer to peer network model, but this time, with a priority on data security and encryption.


Client Side
Server Side
Server
User browser
Client
MainApp
Login Server
Communication
Profiles
Users
Login
Database