Serena Toon

Project B Final Report

COMPSYS 302

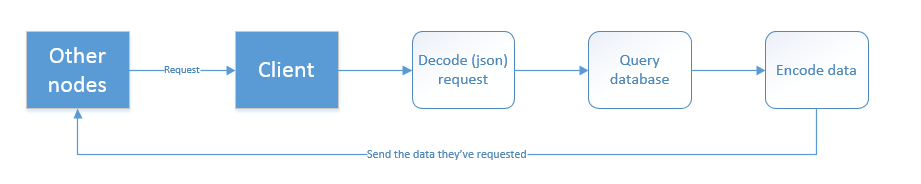
# Overview

The developed system which has been implemented is a functional prototype of a peer-to-peer social media network. It has met the specifications of allowing clients to log in to the system, sending and receiving files to/from other clients, and maintaining client profiles.

# Top-level view of system



At the entry point of the application, the user is prompted to log in to the system. On submission, a request will be sent to the login server. Once authenticated, communication is then done between clients directly.



All data transferred between nodes is JSON-encoded. The basic process of getting data from another node (e.g. getting their profile, status) involves sending a JSON request which will invoke their respective ‘get’ API.

# Significant issues

Several issues were encountered in the web-development aspects of this project. It was hard to develop a system with good UI/UX. In particular, I wanted to have dynamically-refreshing content without refreshing the entire page, which would detract from the user experience. This was overcome by utilising JavaScript/jQuery. Having web-development experience in only HTML and CSS, JavaScript and jQuery were difficult to wrap my head around. The integration of all of these parts (HTML, jQuery, Python) was especially difficult. It was especially confusing how to get jQuery/HTML to call a Python function. The CherryPy framework proved to be especially helpful, as it handled a lot of the requests and simplified the networking side of things a lot. Overall, it helped that with web-development, your changes can be seen live. This allowed for easy trial and error.

# Features

On top of the minimum, required features, there are several features which improve the functionality of the system. Namely, the interactive UI allows most actions to be done on one page; the user does not need to click around and navigate to other pages often. Furthermore, the application will regularly authenticate with the login server once logged in, so that the user’s session will not expire.

# Protocol

The developed protocol included many features to improve security. This made it especially suitable for the proposed system. When developing the protocol, several concerns had to be kept in mind. There was a balance that had to be found between security, reliance on server, redundancy, and difficulty of implementation.

With the proposed use-case where security is of utmost concern, P2P is a very suitable method of communication. Concerns are largely of the company’s server containing sensitive data being hacked into. With P2P, there is no central server to be hacked into. All data is stored locally on the relevant clients’ local machines. Having no reliance on a central server meant no maintenance cost of a server for the company. If relying on a central server, server downtime would mean an outage of all communications. With P2P, this is circumvented.

Considering purely communications, a pure P2P system would be relatively easy to implement at a small scale. Once security is taken into account however, complexity is increased. Adding a login server for authentication greatly increases security and improves the scalability of the system. If a new employee joins the company, they simply need to be assigned a login ID and password, and they can now communicate with other users.

# Development tools

### Python

Python was the language of choice for this application. It is a high-level and weakly-typed programming language, which is easy to learn and use. With the given time constraints and former programming experience with other languages, it was a suitable language. Python also gives us access to the CherryPy framework, which hugely simplifies networking.

### CherryPy

The CherryPy web framework greatly simplifies networking development. It allows us to easily build web applications. It has exhaustive documentation to make it easy to use. Furthermore, it has security features which can be used to make the application more secure.

### SQLite

The database management system used is SQLite. It is very widely-used in the industry, and thus has bindings to many programming languages. Relevant to this application, it has bindings to Python. It is dynamically and weakly-typed, which made it easy to learn and pick up. It does not require any configuration and is portable, simplifying databases a lot for this application.

### HTML/CSS

The frontend has been developed with HTML and CSS. HTML is used to display the desired content on the browser, while CSS styles the content. HTML5 also allows for in-line media players.

### JavaScript/jQuery

JavaScript has been used in order to make the webpage dynamic and interactive. jQuery makes it easy for all interactions to be done on one page, improving the user experience.

# Future development

There is a lot which could be added to this application. At its current state, there is not much done for client-side security. Security APIs could be implemented, as well as adding onto existing APIs with its higher-security parameters.

Currently, there is no implementation of the login server outage protocol. This means that in the event of a login server outage, communications will be brought to a standstill (if not already authenticated). Since there is an existing API for this scenario, it could be implemented in the future.

To add upon its functionality as a social network, an events co-ordination system could be added with the existing API.