COMPSYS 302

Project B: Python

QUENTIN HENG
775834627

Table of Contents

1. Project Brief

Our client is a CEO looking to maximise the potential of his company. However, technology is rapidly growing to a state where they can be considered a threat to confidential information. The client would like to minimise the risk of information leaked outside the company by setting up a social network for employees of the company to communicate with each other.

We have developed hybrid system network where users find and communicate with other authorised users in the network. The core communication of the system is executed through a peer-to-peer(P2P) network. Once authorised, a user has its own node and can communicate with another node directly using an agreed upon protocol used to develop the system. Authorisation is done through a login server which manages users joining the network and finding other authorised users.

The developed system will give the employees of the company a platform to communicate all necessary information with each other. The system provided means that third parties will have more difficult monitoring the company activities thus provide a securer infrastructure to further push the company forward.

2. Technology Stack

Our system is built with python using the CherryPy framework to manage the web server application. All incoming and outgoing connections are processed through an agreed upon application protocol interface(API) where the server can connect with other nodes. All incoming data is stored in a SQLite database which can be accessed to send information out quickly. When accessing the server, CherryPy will serve the browser HTML templates which is configured using Cascaded Style Sheet(CSS) and JavaScript(JS). The CSS modifies the look of the webpage helping with user interface whilst the Handlebars.js maps the content into the HTML template and display it to the end users. Asynchronous JavaScript And XML(AJAX) calls send and receive data from the database in the background to create dynamic webpages.

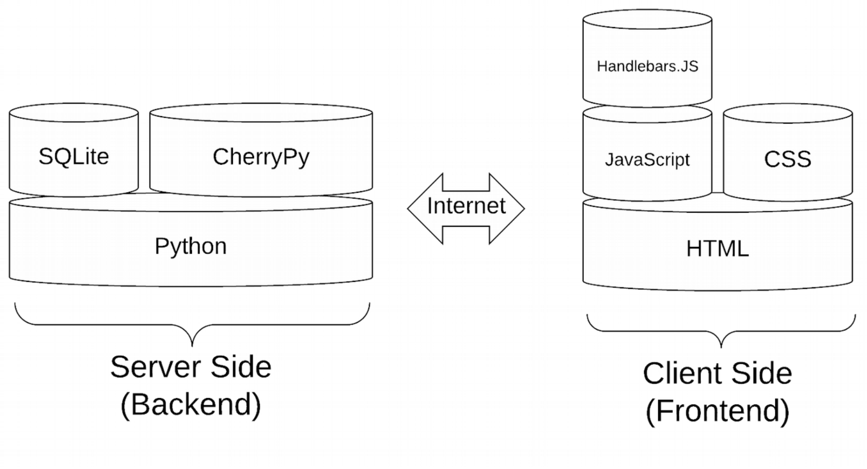

Figure 1: Diagram of Technology Stack

3. Development Issues

3.1 Time & Experience

This project involved unfamiliar elements in which prior experience in full stack development was lacking. The development tools utilised were ideal for the desired application but required extra resources to apply them which caused some issues during the development issues. Though these elements made sense conceptually, implementing them was more difficult than originally intended. In terms of the backend, I had to learn how to implement APIs through the CherryPy framework and implement a relational database using SQLite. For the frontend, I had to learn how to send and update information of the webserver in a browser using HTML, CSS and JavaScript. Despite the restricted timeframe and lack of experience, these issues were overcome by focusing on delivering the key functionality of the system which fulfils the clients use cases.

3.2 Testing

The testing of the backend was difficult due to the nature of networks. I encountered issues with port forwarding which meant I could not test any API that I was hosting. Another issue was some of the extra features required another node to support it. This causes issues when testing since both developers need keep up with the other during the development stage otherwise testing would be delayed.

4. Features

Aside from the core communication functionality, other features include but are not limited to an embedded view, user-friendly interface and serverless database.

4.1 Embed Viewers

The web server features embedded viewers for files sent and received by users. The base application allowed for file transfer but required the user to download find the specific file from a directory then open using a native viewer. The embedded view simplifies the process for the user by allowing them to view all files through the webpage as they are sent as over. The embedded viewer supports a wide range of files which could include files, images, audio and videos. For example, images sent can include but are not limited to png, jpg, jpeg formats. The file is still downloaded to a specified directory if the user prefers a native viewer.

4.2 User Interface

Though not the focus of the prototype, a user-friendly interface was developed to aid users with communications and simulate a platform similar to that of a social media network. While functionality is important to any system, a good user face is key to delivering a streamlined experience for users. A good user interface will abstract the unnecessary details of the system from the user so it can perform its core functionality without difficulty. The interface designed focuses on delivering the key information clearly to the users. All messages and files are separated by the person messaged so it is easy to keep track of conversations with specific users. Each message and file is timestamped and sorted chronologically, to keep track of old conversation history.

4.3 Databases

An serverless database is used to store all necessary information of the users. By storing the information in a database, user information can be stored and obtained with minimal delay from the server to the client. A common approach to data storage would be to store the data in a server so that all nodes can access the information if needed. However, in our approach, only relevant information related to the user is stored. User A does need to store the private conversation between User B and C, in their own node. For the outlined use case, data does not need to be shared as it can raise security issues. Instead, our particular approach closely follows the P2P network model. The database also supports offline use for the system. The conversation history for different users can be viewed whether they are online or not.

5. Network Architecture

In a central server model, the dedicated server manages the data and access of all client nodes. For larger scale operations, the client/server model would be appropriate considering there are many nodes so data would be managed better as a whole rather than scattered. Due to this, this model is also more stable. All data can be backed up so that if one serve goes down, client nodes will still be fully functional. Though data is easily managed, it also poses a security risk since everything can be breached at once. An example would be the Yahoo data breach where 3 billion user accounts were compromised.

For P2P models, data and communications can happen between the clients' nodes. Since each node can act its own server as well as client, this means that this model is inexpensive relative to a client/server. Each node maintains its own data so security breaches are not fatal to the entire system. However, since there is no central server managing the access of the nodes, security must be implemented from the node which is typically harder to implement. Aside from security, the over-reliance on other nodes can cause issues. Unless the clients are updating their databases with other nodes, if one node goes offline, the other clients will be unable to reach that node's data.

For our application, we have used a hybrid system between the two which consists of a login server to manage the user access instead of a pure P2P network. The advantages of this model suit the use cases outlined. The model will be relatively cheaper to implement and a strict protocol can be followed to ensure data is secure and archived correctly. The scope of the use cases does not warrant a steep investment into maintaining a central server infrastructure.

6. Protocol

The proposed protocol supports the necessary use cases identified for this application. Data transferred between nodes are formatted using JavaScript Object Notation(JSON). JSON is a format which makes sending and receiving of data between the nodes easy since data can be easily formatted and parsed. Passwords must be hashed with a salt to protect the users. Other security protocol such as encryption were optional and are subject to specific node implementation if they found the current security standard was not sufficient.

7. Sustainability of Development Tools

7.1 Python

Python is a high-level programming language that can be used for full-stack development. It is relatively easy to learn due to the detailed documentation available. It has a wide range of available packages and libraries that can be imported easily which make product development much easier. In our particular case, web frameworks such as CherryPy allow us to focus on other backend functionality instead.

7.2 JavaScript & Handlebars.js

The application of JS will be vital for future iteration of the prototype. JS is widely used in web development. It allows us to create dynamic content by manipulating our HTML and CSS. This means we can create better interfaces that help us fulfil the use cases.
Handlebars.js is a templating engine which allow us to change the content of our web page using variable data from outside sources such as our database. It is effective tool for templating as it allows us to separate the HTML templating from the JS.

7.3 SQLite

The use of SQLite is appropriate for a P2P application. SQLite is serverless and easy to setup which will allow the nodes to store and read data quickly. For a P2P network, a light-weight database system is ideal.

8. Future Improvements

In future iterations, I would like to implement a stricter security protocol. This would include encrypting data internally before storing it. This would make user data more secure in the case of a breach since the attackers will need to decrypt the data. Some of the higher encryption standard will also prevent users from simply brute forcing the data.

Another feature I would like to implement is a fall-back pure P2P system. Currently, our model relies on the login server to be active at all times in order to gain access to the nodes. There have been cases during development where developers have almost used up the available bandwidth of the server. Having a fall-back system will allow the nodes to communicate with each other even if the server goes down.