Social Network - P2P Server

# Abstract

The aim of the project is to develop a social network application. This technical report outlines our client’s requirements, features, top-level view of our game, significant issues we encountered, suitability of tools, peer to peer methods, protocol, and future improvements of the application. Attached in the appendices are images of the application

# Requirements

Determining a project’s success is dependent on achieving client satisfaction. The client is key in defining and achieving project success. We must gather the client’s requirements and meet their requirements to achieve client satisfaction. The client had 5 minimum requirements.

The application has met the minimum requirements. As the application includes a way to login into the system, the system can automatically find other users on other IP addresses using the data from login server. In the application the user can create and maintain a simple profile page, and the user can send messages, and files to each other. However, to improve the application I have also added features such as encryption, nice UI, two-factor authentication.

# Top Level View

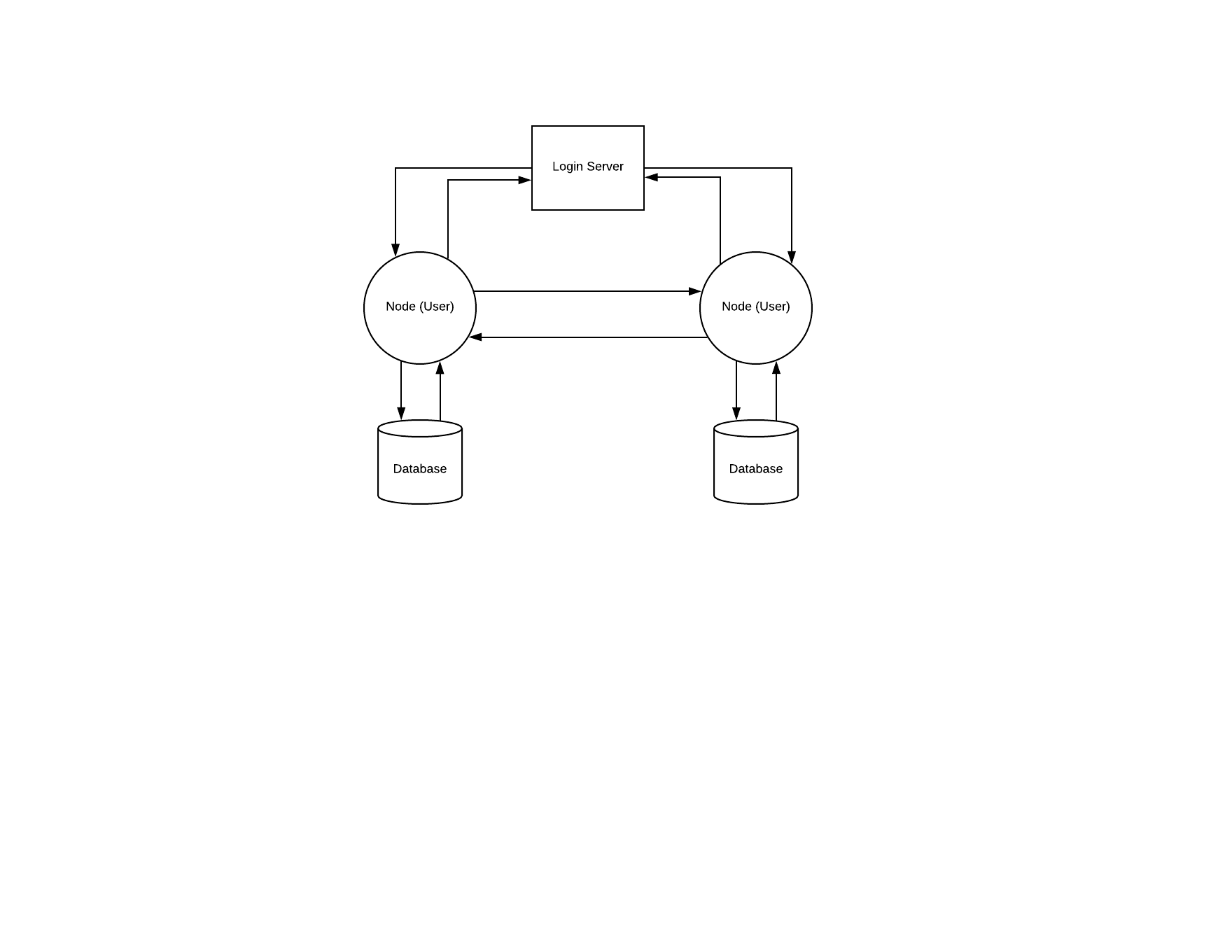
The system includes three main components, the login server, individual application, and individual database.

Figure 1: Diagram of Top Level View

# Features

**Two-factor authentication**

The application has two-factor authentication. This feature creates another layer of authentication when logging into the system. Therefore, the application is more secure.

**Encryption**

The protocol allowed for variation in encryption and hashing standards. I have implemented all the hashing standards and have implemented encryption standards 1-4. Therefore, the application supports XOR, AES, RSA, AES with RSA encryption standards and SHA256, SHA512, bcrypt, scrypt hashing standards. Therefore, communication data is encrypted to the highest possible standard of the destination node, thereby securing communication data which a requirement of the client.

**Nice UI**

The application has a nice user interface. This feature makes it easier for the user to navigate and use the system. Thus, improving the user experience.

**Embedded Viewers**

The application supports embedded viewers of video, audio, image files. This feature makes the user experience more user-friendly and enjoyable because you can view the files within the web-app instead having manually download the file to view its contents.

# Significant Issues

A major issue was the limited time we had to create a minimal viable product (MVP) for our client. A significant amount of time was spent debugging and fixing issues within the application. This is because a lot of bugs were caused by human error and not due to logic flaws. For example, this included assigning a ‘y’ variable to a ‘x’ variable due to copy and pasting the previous lines of code, while forgetting to edit the variable name in the new lines of code. These bugs were hard to notice within source code.

Due to a large amount of time being unexpectedly spent on debugging, a multitude of features that could not be delivered by the deadline such as fall-back P2P methods, offline, group messaging. Therefore, we had to rescope our project and decide to not include some of the features in our MVP.

# Suitability of Tools

CherryPy is suitable tool as it is a minimalist python web framework. CherryPy allows us to quickly implement a webserver. SQLite3 is suitable tool for this project as it is a SQL database engine that allows us to store all our data. Bitbucket is a suitable git version control tool. We can track all our file and its changes over the course of this project easily. Projects of this scale are typically iterative thus the ability to track these changes and refer to them is valuable. This is especially important in the case that we want to rescope as git allows us to find the version that we can want to rebase to.

# Peer to Peer Methods

In a central server model all the traffic goes through the central server, this allows for full storage of messages for offline users and easy monitoring of user permissions. However, a central model would be very costly. In a pure peer to peer model, all traffic goes from peer directly to the destination peer. This means there is no central server that could easily be compromised, instead an attacker must attack every peer to gain access to all the data in the network. The major issue with a full peep to peer model is that finding the IP addresses of peers is difficult and authenticating users into the system would also be challenging. Both these models have their own benefits and disadvantages, in our project we decided to use a hybrid system that utilises a login server. The central login server authenticates users and maintains a list of every node’s IP address and port. Therefore, data between nodes can remain peer to peer. A major disadvantage is that if the login server were to go down, the whole system would not work as the system relies on the login server for collecting every node’s IP address and port.

# Protocol

The protocol used in this project was suitable for this application. All messages and data transferred between nodes were JSON encoded to make parsing and processing the data easier. The protocol was flexible which allowed for all the students in the class to include the parameters they would like to implement.

# Future improvements

All the features outlined weren’t completed by the deadline. Therefore, additional features and improvements can be implemented in the future to further improve our product.

Improvements can be made on this product as this product is only a prototype. These include having a ability to not rely on the login server and use full per to peer methods. Also, another improvement can be made so the user can offline message people. Group conversations can also be added, which would allow users to have group messages.

These additional features would greatly improve the user experience. Thus, the user is can easily use the system and would enjoy the user experience much more.

# Appendix

## Appendix A: Images Of Application

