Python Report

# Abstract

The aim of this project was to build a peer-peer network to increase the security of the company by ensuring rival companies couldn’t access and steal private data. The main programming language python java but parts of Html, CSS and JavaScript were required along the way. The project was made to be challenging as most students had little to no experience in python. However, after researching and trying different implementations we were eventually able to make a working interface.

# Meeting Requirements

I believe the system has checked all the requirements to be suitable for use in the company. The service allows communication between peers using a login server. It is secure ensuring only select members (employees) have access to the system by requiring them to use their username and password. All online users are displayed on the main menu snsuring each employee knows who they can interact with. Users can message one another to inform that person of an important task that needs to be completed. Files upto 5Mb in size can also be sent; this includes images, videos, docs and many more. To make the system more personal, each user has a bio which they can edit at their pleasing.

Almost all data is backed up to databases ensuring that nothing is accidentally deleted and all records can be tracked back to.

# Notable Features

The highlight features include a simple GUI. It was designed with simplicity in mind to ensure users can see the information they need to see and can get quick access too any tools at their disposal. All of the main features can be accessed through the main menu.

# Top-Level View

# Peer-Peer Overview

# Protocol Overview

The protocol went through careful consideration to achieve a balance between functionality and ease. It manages to keep things relatively simple but still allows for all necessary conent to be sent and delivered.

# Tools

The main software used was python, a interpreted high-level programming language. Python was a good choice as it allowed for readable code that was easier to code (requiring less lines). Python is an interpreter meaning there was no need to compile. This was useful as continuous testing could be done without much delay.

Other software used included html, css, jinja2 and a little javascript. Html and CSS were essential to make the user interface simple to navigate and access. Jinja2 helped allow content to easily displayed where required.

# Future Improvements

Despite the extension provided, a lack of time was the main culprit behind not completing some of the more complex tasks. If time wasn’t an issue, some of the more impressive features such as multiple sessions and two factor authentications may have been implemented. As the the aim of the project was to provide a functional back-end, the amount of effort put into the front-end wasn’t nearly as much. This resulted in the server lacking an aesthetic look. With more time and effort a much cleaner and functional user interface could be provided.