Activity Breakdown (primary activities, hours spent)

Each member of your team should contribute a short description (no longer than a paragraph) of his or her primary activities during this sprint, including an estimate of how many hours he or she worked. (Don't be afraid of reporting numbers that seem low. Your grade will not be affected by this number. We promise.) Each member should also assess whether these activities were a valuable use of time. If not, what could he or she have been doing instead?

Brandon:

I worked mainly on implementing the secure channel between the server and the client as well as implementing a more efficient and extensible file transfer framework. The code was also tested thoroughly to ensure that it was suitable for merging with the rest of the team's code. The time spent on this was productive as it helped to improve the core functionality of our product and also introduced additional security functionality. However, I think I should learn more about SQL functions to help out the team during testing. I spent around 20 hours.

Louise:

Together with Ruixin, I did server-client communication. Specific tasks I did include encryption for files and users, protocol design (structure of JsonObjects between server and client), ServerView, threading for the Client, merging Brandon's code into the server and client. I also setup the maven build system. I spent 20+ hours.

Most of the time spent was fairly productive, although I felt like I wasted a bit of Ruixin's time because of the changes the encryption caused to the database. I should also have thought through threading more carefully, because I had to/have to rewrite all the calls of Task to be in the Views and not in the Controllers.

Ruixin:

In this sprint, I focused on coding the sql queries and connecting the client to the server by coordinating the Json packets exchanged. I also worked on designing the database models and relations, and designed some protocols with Louise. It was a generally good use of time, although I feel like I should have tested some queries more before writing more as there were some similar errors that propagated across the methods. I also should have planned out the protocols more before diving into writing the functions. Total time spent was about 20 - 30 hours.

Zilong:

I worked on unit testing and debugging functions in charge of filtering valid passwords, usernames, email addresses etc. In addition, I worked with Brandon to test out the file transfer capabilities of the SSL code, which was a major improvement from our unsecured version in the previous sprint, before doing integration testing with Ruixin and Louise's components. As this part can be done relatively independent with respect to the other components of the project, it was relatively productive. I do feel that I should have tried to test out the SQL functions, though my inexperience in such cases slowed the team down. I spent around 15 hours.

Productivity Analysis

Compare the plan for your sprint to what really happened. Did you do what you planned, or did you do something else? Did all of your chosen system backlog items get finished? What took more time than you expected? What took less time than you expected? Most importantly, how will you change the way you work in future sprints?

We managed build a minimum viable product that included all functions that we deemed to be of highest priority. Nonetheless, there are some things that we wanted to achieve but were not able to integrate by this sprint:

- Being able to view the file and user logs. Currently, the information is stored in the database but have not written code to access it and display it in a user-friendly manner.
- A notification based on the number of failed login attempts
- Rate-limiting of logins (although this is partially achieved through a slow password hash by increasing the number of iterations to 10000).
- More extensive unit testing, especially of the database.

We will definitely prioritise unit testing in the future and consider bottlenecks more carefully. This sprint, we were often bottlenecking each other, resulting in a lot of periods where individuals were writing code that they could not test because it relied on someone else's code that was not completed. This inability to test cascaded down to an inability to correct design decisions, leading to many expensive ramifications down the road such as having to recreate all the SQL tables several times because of columns that needed to be added.