Workflow Separation

We worked in pairs: server-client communication+SQL (Ruixin+Louise), SSL+unit testing (Brandon+Zilong). Each pair tested and reviewed each other's' code. We occasionally did pair programming.

Testing of Source Code

A portion of the source code is unit tested, specifically:

- cs5431_client.util.Validator

Which ensures that usernames, passwords, email addresses, filenames, IP Addresses are valid, and do not contain illegal characters (like “/” or “.”) and fulfill certain length requirements. While some tests may not really be needed, as they are simple boolean checks on length, others which include using methods invoked from pre-implemented functions, such as IP Validator and Email Validator from Apache Commons required some amount of testing.

As this part is very important wrt sanitizing user inputs, unit tests for this class was prioritized as of now. In addition, as this part does not bottleneck the other parts of the project, testing was done for this part.

Example of testing for valid email addresses:

```
@Test
void test_validEmail_notLegit(){
  boolean noAt = Validator.validEmail("bcp39atcornell.edu");
  boolean noDot = Validator.validEmail("asdasd@gmailcom");
  boolean AtInName = Validator.validEmail("John@Doe@gmail.com");

  assertEquals(false,noAt);
  assertEquals(false,noDot);
  assertEquals(false, AtInName);
}
```

We performed manual correctness testing on the following components:

- Login
- User registration
- Users navigating through the file system
- Users uploading files and creating folders
- Users changing passwords

We ensured that the files and folder names created are encrypted. We also ensured that the passwords that are transmitted to the server are not in the plain, as this can be used to decrypt the user’s private key.

We plan to do unit testing on these parts as well once we familiarize ourselves with how to do unit testing on databases with things such as DBUnit etc.

FindBugs
FindBugs was performed on the almost final product, minus a few bugfixes. FindBugs found 165 bugs, of which none were scary nor troubling

Scary bug (rank 8):
Read of unwritten field logEntries in org.cs5431_client.model.FileLog.addLogEntry(FileLogEntry)
Troubling bug (rank 12):
Unwritten field: org.cs5431_client.model.FileLog.logEntries

These two bugs were rectified by initialising logEntries to an empty list. Furthermore, since we have not properly implemented file log viewing due to decryption woes, logEntries is never read in our code.

There were another 2 bugs labelled Malicious Code Vulnerability

Rank 18:
org.cs5431_client.model.FileSystemObject.getLastModified() may expose internal representation by returning FileSystemObject.lastModified

Rank 18:
new org.cs5431_client.model.FileSystemObject(int, String, Folder, Timestamp) may expose internal representation by storing an externally mutable object into FileSystemObject.lastModified

These was rectified by performing deep copies of the timestamp lastModified.

24 bugs are internationalization bugs, mainly caused by us assuming that the charset is UTF-8. However, these are false alarms since we never display the variables concerned to the users - the alarms are sounded on fields such as IVs and salts.

32 bugs are performance bugs, some of which we fixed. For example,
Rank 16:
Boxing/unboxing to parse a primitive
org.cs5431_client.controller.AccountsController.getFolderFromId(int, int)
Was fixed by changing Long.valueOf to Long.parseLong.

42 bugs are bad practice bugs, most of which are about naming. These will be rectified by the next milestone as some carefulness is required to make sure the renaming is consistent.

12 bugs are experimental bugs, all of which are either due to failures to close streams or sql ResultSets. However, some of the streams are not supposed to be closed, such as streams listening for multiple json strings.

52 bugs are dodgy code bugs, many of which are caused by dead stores since we have not implemented the full functionality and we store many values without accessing them.