Preliminary Design

1. Class:

   User
   - internal user, user ID must be unique.
   - external user(hacker), don't have user ID I guess.

   Group
   - group ID must be unique.

2. Use a simple MySQL database for information management.
3. Users that are in the same group can change to each other's local directory but all files in the directory will be encrypted and the SFS should prevent the user from opening files that do not belong to him.
4. Users that are in different groups can NOT change to each other's local directory.
5. External users would have the ability to see files from all groups but they are encrypted.
6. External users would have the ability to modify the contents of files.
7. After a file is modified by other users, the owner of that file should be notified when he logs in the SFS.
8. The owner should have the ability to remove all changes that are made by others.
9. We don't need to use the Cybera at the first place
   The plan is as following:
   A. we create server and client classes and connect them using a local port (run both server and client on the same computer and make sure they work properly).
   B. then we move to cloud service.
10. Best encryption method for a file is AES and best encryption method for password would be RSA, we'd need to decide whether to use both of them or one of them.
11. https://pypi.org/project/rsa/
12. https://pypi.org/project/aes-cipher/
13. https://stackoverflow.com/questions/5278759/how-to-store-txt-files-mysql-database
14. Q6: How do we know who a user is? Do we pre-load information into the database, or do we allow anyone to register for an account & login?

    Answer: After you run the server, that program should be able to receive commands from the command line. You have to define some commands to create a new user and password (if the username is repetitive, it should return an error indicating that a user with this username exists.), create a group(no repetitive groups!), add existing users to existing groups(no duplicates!), delete a user (error when asked to delete a user that does not exist).

15. No need for multithreading server
16. The owner should be able to share a file.

17. Database schema
    a. 1st table, registration info, user id, password and group ID
    b. 2nd table is for directory, user ID and directory ID. We need to keep track of the parent of each directory. File ID, unique ID with each file. File name in the second table. Permission field.
       User ID, directory ID, permission.
    c. 3rd table, FIle ID PK, name of file, parent directory ID.

1. User ID, Group ID, password.