SFS Design Document

Communication protocol:
Requests would be a string separated using space.
Request format —> int(operation) data1 data2 ….
E.g. 0 username password
Explanation: 0 is the code for login operation, username and password will not be encrypted at this stage (client to server)
Server sends ACK back to client to either confirm that a command has successfully been executed or if an error has occurred
Format: Use int codes for each possible response (0 for success)

4 levels design

1. Common home directory
2. Personal home directory
3. Custom directory
4. files

Client Side:

Current location(directory) will be stored as a private attribute in the user class.
File encryption shall be applied on the client side.

1. ‘login’. Login will be the first operation that every user needs to do. Users would enter their username and password and they shall not be encrypted on the client end.
   Code for ‘login’ is 0 and sample request: **0 username 1234567Alberta%**
2. ‘sign up’. Sign up is also a start operation that every user needs to do in order to use SFS. The request is similar to the ‘login’ operation.
   Code for ‘signup’ us 1 and sample request: **1 username 1234567Alberta%**
3. ‘ls’. List all directories in the home directory.
   Code for ‘ls’ in home directory is 2 and sample request: **2 username**
4. ‘ls’. List all files in the selected directory.
   Code for ‘ls’ in selected directory is 3 and sample request:
   **3 username current_directory_name**
5. ‘cd ’. Change directory operation.
   Code for ‘cd’ is: 4 and sample request: **4 username directory_name_change_to.**
6. ‘mkdir’. Create a new directory.
   Code for ‘mkdir’ is 5 and sample request: **5 username new_directory_name**
7. ‘cat’. Display contents of a file.
   Code for ‘cat’ is 6 and sample request:
   **6 username current_directory_name file_name**
8. ‘check integrity’. Check integrity of files and folders once a user login.
   Code for this operation is 7 and sample request:

7 0 username directory_list
Server check if directory list matches with what we have in database, if yes send confirmation key back to client.
7 1 username directory_name file_list
If match, continue
7 2 username directory_name file_name file_contents
If match, continue
7 2 username directory_name file_name_2 file_contents

To sum up, 7 0 is for validating the directory list. 7 1 is for validating file list in a directory and 7 2 is for validating the contents of a file

9. 'add'. Add contents to a .txt file.
   Code for this request is 8 and sample request is: 8 username directory_name file_name new_contents.

What you should do for existence/permission check:

E.g. user sends me a create file request

1. Fetch all files and directories that the user owns or has permission.
2. Generate a look-up table with such format {key: encrypted absolute path, value: (encryption key, encrypted file name)}
3. Pass the look-up table to a function in file manager class called:
`getFileListInCurrentDir()`
This function helps you to create a subset of the look-up table which only contains the files in the current directory. We'll call it the new look-up table
4. Now, you will need to iterate through the new look-up table and decrypt file name, then you'll need to compare the decrypted filename with the filename you received from the user, if they match then the file already exists. If not then we should be able to create the file.

Login : 0 username password
Signup : 1 username password
Create file: 2 filename
Display file contents: 3 filename
Add file contents: 4 filename contents
Create directory: 5 directory_name
Change directory: 6 directory_name

List Dir: 7
Rename: 8 oldname newname
Give permission: 9 target_user filename
Remove permission: 10 target_user filename