There are three main files as asked for

metaserver.py
dataserver.py
filesystem.py

Step 1. to start all servers

```
cd fusepy
python metaserver.py 51234
python dataserver.py 51235
python dataserver.py 51236
python dataserver.py 51237
python dataserver.py 51238
```

Step 2. Start client

```
python filesystem.py fusemount 3 3 51234 51235 51236 51237 51238
```

Step 3. Mount the filesystem on fusemount and see functioning in step 1 and 2

```
cd fusepy/fusemount
touch my.txt
echo "hello" > my.txt
cat my.txt
mkdir a
cd a
echo "hi" > you.txt
```

(all these commands also given in test.sh)

Step 4: Corrupting one of the server data

implemented in " def corrupt(self,key): " in dataserver.py

Run commands in corrupt.py. We can make the rpc call to the server we want to corrupt.
The key to be corrupted is also provided in corrupt.py and we corrupt it by changing its value in the server. The key is '/a/you.txt&&data'. It returns 1 if it is successfully corrupted.

To handle this corruption we have implemented quorum logic on client side by maintaing server flags in a list called 'set'.
'1' represents correct server data
'0' means corrupted server data

If '0' is printed then the wrong server repairs itself from a replica that has the correct data if Qr >= 3.
This functionality is transparent to the client and the client never sees if data was corrupted as whenever we open the file, it has the correct data taken from replicas.

repair mechanism implemented in "def repair_corrupt(self, url): " on server side.

Step 5: Crashing server

implemented in def terminate in dataserver.py or we can corrupt manually by pressing Ctrl+C

The crashed server is able to restart empty and can take backup from its peers.
Suppose dataserver with port 51237 crashes then run the following command with all remaining url's:

python dataserver.py 51237 51235 51236 51238

Functionality implemented in polling(self) and backup_from_peer(self, url) in dataserver.py

To pass correct url to backup_from_peer we have applied quorum and done polling on the server side too by again using the method of setting flag status to '1' for correct servers in the method polling(self).

Additionally performance_script.sh and Replica_FS_tests.sh can be used to test the functionality and performance of the file system.