­­­­­­Robert Judka

CS550

Programming Assignment 1

Verification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Scenario | Test Steps | Test Data | Expected Results | Pass/Fail |
| run server | 1. run ‘./indexing\_server’  2. run ‘lsof -i :9999’ in Linux shell | N/A | server listening on port 9999 | P |
| run peer | 1. run ‘./peer peers/p1/’  2. enter ‘l’ into cli | files in ‘peers/p1/’ | indexing server lists all files in ‘peers/p1/’ | P |
| add new file to peer | 1. add new file to ‘peers/p1/’  2. enter ‘l’ into cli | files in ‘peers/p1/’,  file ‘foo.txt’ | indexing server lists all files in ‘peers/p1/’, including ‘foo.txt’ | P |
| delete file from peer | 1. delete file ‘b.txt’ from ‘peers/p1/’  2. enter ‘l’ into cli | files in ‘peers/p1/’,  file ‘foo.txt’ | indexing server lists all files in ‘peers/p1/’, not including ‘b.txt’ | P |
| existing file search | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli | files in ‘peers/p1/’ | output listing current peer owning ‘a.txt’ | P |
| nonexistent file search | 1. enter ‘s’ into cli  2. enter ‘foo’ into cli | files in ‘peers/p1/’ | output stating file not found | P |
| file download from current peer | 1. enter ‘r’ into cli  2. enter peer’s current client id into cli | files in ‘peers/p1/’ | output stating no retrieval perform because the peer is the current client | P |
| run 2 peers | 1. run ‘./peer peers/p1/’  2. run ‘./peer peers/p2/’  2. enter ‘l’ into either cli | files in ‘peers/p1/’,  ‘peers/p2/’ | indexing server lists all files in ‘peers/p1/’ and ‘peers/p2/’ | P |
| existing file search for file owned by other peer | 1. enter ‘s’ into cli  2. enter ‘k.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output listing other peer owning ‘k.txt’ | P |
| existing file search with both peers sharing that file | 1. enter ‘s’ into cli  2. enter ‘j.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output listing both peers owning ‘j.txt’ | P |
| existing file download from other peer | 1. enter ‘r’ into cli  2. enter other peer’s client id into cli  3. enter ‘k.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output showing original name of file downloaded and the name of the new file (both are ‘k.txt’) | P |
| nonexistent file download from other peer | 1. enter ‘r’ into cli  2. enter other peer’s client id into cli  3. enter ‘foo’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output stating other peer does not have file | P |
| existing file download from other peer with both peers sharing that file | 1. enter ‘r’ into cli  2. enter other peer’s client id into cli  3. enter ‘j.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output showing original name of file downloaded and the name of the new file (new file with name ‘j-origin-{other peer’s client id}.txt) | P |
| file search while other peer making sequential requests | 1. run script that loops other peer making search requests  2. enter ‘s’ into cli  3. enter ‘k.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output listing other peer owning ‘k.txt’ | P |
| file download while other peer making sequential requests | 1. run script that loops other peer making search requests  2. enter ‘r’ into cli  3. enter other peer’s client id into cli  4. enter ‘k.txt’ into cli | files in ‘peers/p1/’,  ‘peers/p2/’ | output showing original name of file downloaded and the name of the new file (both are ‘k.txt’) | P |
| 10 peers all making 500 sequential file search requests | 1. run ‘python peer\_simulation.py 10’ | files in ‘peers/p1/’,  …,  ‘peers/p10/’ | logs showing 500 sequential start/end search requests for each peer | P |
| peer quitting network | 1. enter ‘q’ into cli  2. enter ‘l’ from other peer | files in ‘peers/p1/’, other peer directory | indexing server showing disconnection and cleanup message, lists only files from other peer directory | P |
| killed peer process | 1. enter ^C into cli | files in ‘peers/p1/’, other peer directory | indexing server showing disconnection and cleanup message, lists only files from other peer directory | P |