Testing:

For testing my server, I did each individual part separately. I started by implementing the PATCH functionality which I tested by using curl commands and manually checking the mapping file. After confirming that my PATCH commands were working, I proceeded to implement functionality for being able to GET and PUT using alias names. To test these, I used a flow similar to this:

1) open the server using ./httpserver localhost 8080 -a map.txt
2) run "curl -s -v -T <textfile> localhost:8080 --request-target abcdefghijklmnopqrstuvwxyz0" to PUT the file onto the server
3) create a file on the client called "send" that contains ALIAS abcdefghijklmnopqrstuvwxyz0 alias1
4) run "curl -s -X PATCH localhost:8080 -T send -v"
6) run "curl -s -v localhost:8080 --request-target alias1" which successfully GETs the data from abcdefghijklmnopqrstuvwxyz0 using alias1
7) run "curl -s -v -T <textfile> localhost:8080 --request-target alias1" which successfully PUTs data into abcdefghijklmnopqrstuvwxyz0 using alias1

And I ran tests like these many times with files of varying lengths, names, and aliases. After that, I implemented logging which was tested by simply looking at the generated log file.

Question:

Q. Explain the difference between fully resolving a name (to an httpname) when the name is created and the approach that you're taking for this assignment. Give an example of when it might be useful.

A. The difference between fully resolving a name in this assignment compared to last assignment is that previously, we are given a resource name that is exactly the same as the file name on the server. This was simple because all we had to do was open the file that was specified. The difference this time is that we allowed aliases, so the process in resolving a name in this assignment is that we had to determine if the given name was a resource name or an alias name, then search in the mapping file to find the resource name if the given name was an alias. An example of when aliases are useful would be when the client has no knowledge of the actual file names on the server. Since the client and server are separated, its not guaranteed that the client knows the exact name of the file on the server, but instead, they know an alias. This allows the user to just send the alias to the server instead of wondering what the file name is on the server.

Q.What did you learn about system design from this class? In particular, describe how each of the basic techniques (abstraction, layering, hierarchy, and modularity) helped you by simplifying your design,making it more efficient, or making it easier to design.

A. In this class, I learned how difficult it is to build larger systems. In my previous classes, I heard about abstraction, layering, hierarchy, and modularity but they never really had as big of an impact as it did in this class. Using these techniques, it allowed modifying my httpserver to be significantly easier than it would have been if I hadn't used the techniques. For example, if I did not use these techniques and just wrote everything into a huge block of code, the programming assignments would've been significantly harder to modify because a small change could require changing the entire system/design. Since we used these techniques, modifying the server became easier since we could build each small part separately and put them together like puzzle pieces rather than 1 huge block.