EECS 468 Assignment 6 Testing Procedures

This document details how EECS 468 Assignment 6 was tested for each of the following functionality:

For all of them “curl” was used to send HTTP requests with various test cases through replit’s hosted server and its shell.

GET:

Using “curl” various GET requests were sent to the server with various files pre-placed in the working directory. Here were the requests and what the respone was:

“curl -X GET localhost:8000/example.txt” -> “this is a file” (the contents of the .txt file in question)

“curl -X GET localhost:8000/index.html” -> *the file contents (a longer test html page)*

“curl -X GET localhost:8000/folder/txt.txt” -> *also the file contents*

“curl -X GET localhost:8000/folder” -> “txt.txt” (the only file in the dir named folder)

“curl -X GET localhost:8000/myfile.txt” -> “File not found” (since this file doesn’t exist)

All of these outputted the expected behavior for the GET function.

These commands accurately test the GET method as they handle the situations in which it will be used reading various file types, reading files within deeper dirs., reading dirs., and trying to read a file that doesn’t exist. Since it handled all of these situations with the expected output, we can say that GET passed.

PUT:

“curl -X PUT localhost:8000/myfile.txt” -> *created an empty file named “myfile.txt” in the working dir*

“curl -X PUT localhost:8000/new.txt -d "This will be in the file"” -> *created a file named new.txt with the text “Thi"s will be in the file” in it*

“curl -X PUT localhost:8000/new.txt -d "Not anymore"” -> *changed the contents of the file new.txt to the text “Not anymore”*

Similar to GET these represent the use cases of PUT. Having created a new empty file, a new file with text, and overriding and existing file. These are the use cases for the method and since it had the expected output each time, it passed.

DELETE:

“curl -X DELETE localhost:8000/new.txt” -> *deleted the new.txt file we created*

“curl -X DELETE localhost:8000/folder/txt.txt” -> *deleted the file txt.txt within the folder*

“curl -X DELETE localhost:8000/folder” -> *deleted the dir named folder*

“curl -X DELETE localhost:8000/notafile.txt” -> *did nothing as the file does not exist*

Similar to the previous two, I tested every use case of delete, including deleting a file, deleting a file within a dir, deleting the dir itself, and trying to delete something that doesn’t exist. Since it gave the expected output for each, it passed.

MKCOL:

“curl -X MKCOL localhost:8000/folder” -> *recreates our “folder” dir from earlier*

“curl -X MKCOL localhost:8000/folder” (again) -> *returns nothing as the dir already exists (but also doesn’t crash ☺)*

Here I tested every use case of MKCOL, which was to make a dir and also try to make a dir that already exists. Since it showcased the expected result for each command, it passed.

Error Handling:

“curl -X TRACE localhost:8000” -> “Method TRACE not allowed” (which is what we want to see)

The other error handling methods were showcased in each method’s section if it was an error associated with that method. Since the others are hanlded within each method’s testing, the only things we have to test are the aspects which call the methods. The only real error that can occur here is the user requests a method the server doesn’t support, and therefore since it gave the correct output, it passed.