Matthew Madore

Shane Waxler

Github Link:

https://github.com/sfsu-csc-667-fall-2021/webserver-csc667_server_project-shanew-matthewm

Full names of team members * A link to the github repository * A copy of the rubric with each item checked off that was completed (feel free to provide a suggested total you deserve based on completion)

Project discussion:

Our projection is a demonstration of a simple server using HTTP 1.1 protocol. Our relays messages from the client/browser side to the server side and vice versa. Our project attempts to handle CGI scripts along with not completely handling authorization.

Brief description of architecture:

After building the project with commands javac WebServer.java, java WebServer, you can then open a browser window and go to the url localhost:8080 which will send a GET request to from the client to the server and be handled accordingly. Httphandler.java handles all the different responses and some authorization while Htpassword includes the methods the actually encrypt and decrypt passwords/usernames. Webserver.java is our driver class.

Problems you encountered during implementation, and how you solved them:

We ran into problems with the encryption and decryption and were unable to get a proper decryption. When printing a decrypted message to console, it would print as diamond enclosed question marks. We have little idea why this was occurring other than when diamond enclosed question marks print to screen, you are attempting to print characters that don’t exist.

A description of your test plan (if you can’t prove that it works, you shouldn’t get 100%)

TEST PLAN: type javac *.java and java WebServer

This will start the server and print out which port it’s on.

Grading Rubric:

Category	**Description**		
Code Quality	Code is clean, well formatted (appropriate white space and indentation)		5
	Classes, methods, and variables are meaningfully named (no comments exist to explain functionality - the identifiers serve that purpose)		5
	Methods are small and serve a single purpose		3
	Code is well organized into a meaningful file structure		2
Documentation	A PDF is submitted that contains:		3
	Full names of team members		3
	A link to github repository		3
	A copy of this rubric with each item checked off that was completed (feel free to provide a suggested total you deserve based on completion)		1
	Brief description of architecture (pictures are handy here, but do not re-submit the pictures I provided)		5
	Problems you encountered during implementation, and how you solved them		5
	A discussion of what was difficult, and why		5
	A thorough description of your test plan (if you can’t prove that it works, you shouldn’t get 100%)		

Functionality - Server	Starts up and listens on correct port		

Category	**Description**		
	Logs in the common log format to stdout and log file		
	Multithreading		
Functionality - Responses	200		
	201		
	204		
	400		
	401		
	403		
	404		
	500		
	Required headers present (Server, Date)		
	Response specific headers present as needed (Content- Length, Content-Type)		
	Simple caching (HEAD with If-Modified-Since results in 304 with Last-Modified header, Last-Modified header sent)		
	Response body correctly sent		

Functionality - Mime Types	Appropriate mime type returned based on file extension (defaults to text/text if not found in mime.types)		
Functionality - Config	Correct index file used (defaults to index.html)		
	Correct htaccess file used		
	Correct document root used		
	Aliases working (will be mutually exclusive)		
	Script Aliases working (will be mutually exclusive)		
	Correct port used (defaults to 8080)		
	Correct log file used		
CGI	Correctly executes and responds		
	Receives correct environment variables		
	Connects request body to standard input of cgi process		