

**Akhil Gandu (920128024) &  
Katie Kennedy (920628472)  
September 20, 2020**

**CSC 667/867 Internet App Development  
Instructor John Roberts  
Web Server Project**

<b>Introduction &amp; Github Repository</b>	<b>2</b>
<b>Scope of Work</b>	<b>2</b>
<b>Code Architecture</b>	<b>4</b>
<b>Discussion</b>	<b>5</b>
<b>Problems We Encountered</b>	<b>5</b>
<b>What Was Difficult</b>	<b>5</b>
<b>Testing Process</b>	<b>5</b>

## Introduction & Github Repository

This assignment required us to implement a web server from scratch with Java. Our final code for this assignment can be found under the master branch at the following GitHub repository: <https://github.com/sfsu-csc-667-fall-2020-roberts/web-server-katie-akhil>

## Scope of Work

As shown by the table below, all items and functionalities listed in the requirements document are implemented and running correctly in our final project. Our team suggests full credit for this assignment.

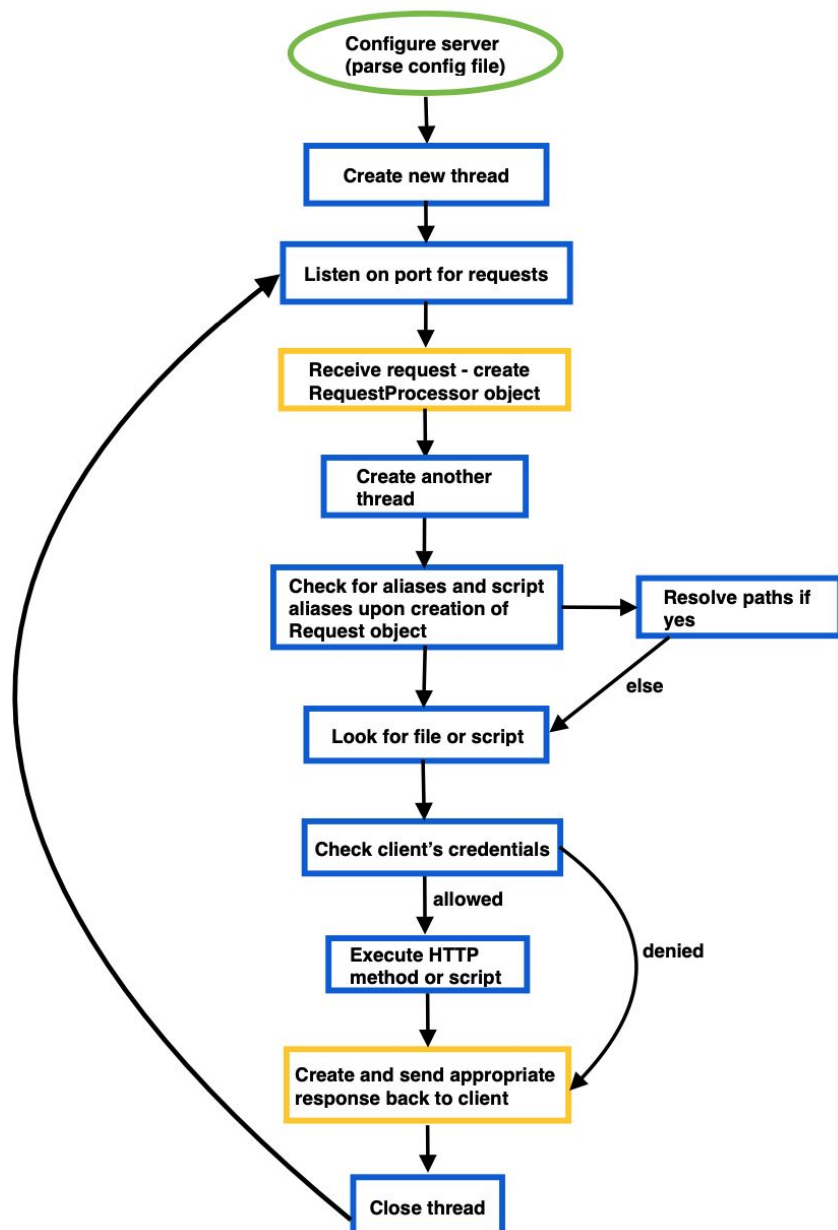
Category	Description	Completed
<b>Code Quality</b>	Code is clean, well formatted (appropriate white space and indentation)	X
	Classes, methods, and variables are meaningfully named (no comments exist to explain functionality - the identifiers serve that purpose)	X
	Methods are small and serve a single purpose	X
	Code is well organized into a meaningful file structure	X
<b>Documentation</b>	A PDF is submitted that contains:	X
	Full names of team members	X
	A link to github repository	X
	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)	X
	Brief description of architecture (pictures are handy here, but do not re-submit the pictures I provided)	X
	Problems you encountered during implementation, and how you solved them	X
	A discussion of what was difficult, and why	X
	A thorough description of your test plan (if you can't prove that it works, you shouldn't get 100%)	X
<b>Functionality - Server</b>	Starts up and listens on correct port	X
	Logs in the common log format to stdout and log file	X

	Multithreading	X
<b>Functionality - Responses</b>	200	X
	201	X
	204	X
	400	X
	401	X
	403	X
	404	X
	500	X
	Required headers present (Server, Date)	X
	Response specific headers present as needed (Content-Length, Content-Type)	X
	Simple caching (HEAD with If-Modified-Since results in 304 with Last-Modified header, Last-Modified header sent)	X
	Response body correctly sent	X
<b>Functionality - Mime Types</b>	Appropriate mime type returned based on file extension (defaults to text/text if not found in mime.types)	X
<b>Functionality - Config</b>	Correct index file used (defaults to index.html)	X
	Correct htaccess file used	X
	Correct document root used	X
	Aliases working (will be mutually exclusive)	X
	Script Aliases working (will be mutually exclusive)	X
	Correct port used (defaults to 8080)	X
	Correct log file used	X
<b>CGI</b>	Correctly executes and responds	X
	Receives correct environment variables	X

	Connects request body to standard input of cgi process	X
--	--	---

## Code Architecture

Our web server makes use of object oriented-style programming in Java to parse configuration files and HTTP requests, execute HTTP methods, and send appropriate responses back to the client. The following graphic displays the relationship between classes and their objects, and shows the flow of execution for our program:



## **Discussion**

### **Problems We Encountered**

One main problem we encountered was trying to split up the work amongst two people. This project's components were not easy to compartmentalize because the system's flow is so linear. This assignment required us to develop almost exclusively during live Zoom meetings with each other to avoid conflicts in structure.

Another problem we dealt with was trying to deal with version control while one team member was using a Mac OS and the other was using a Windows OS. We sometimes encountered issues when pushing different `httpd.conf` files to the same GitHub repository without thinking about the conflicts in a file with the same name. We fixed this eventually with the `.gitignore` file.

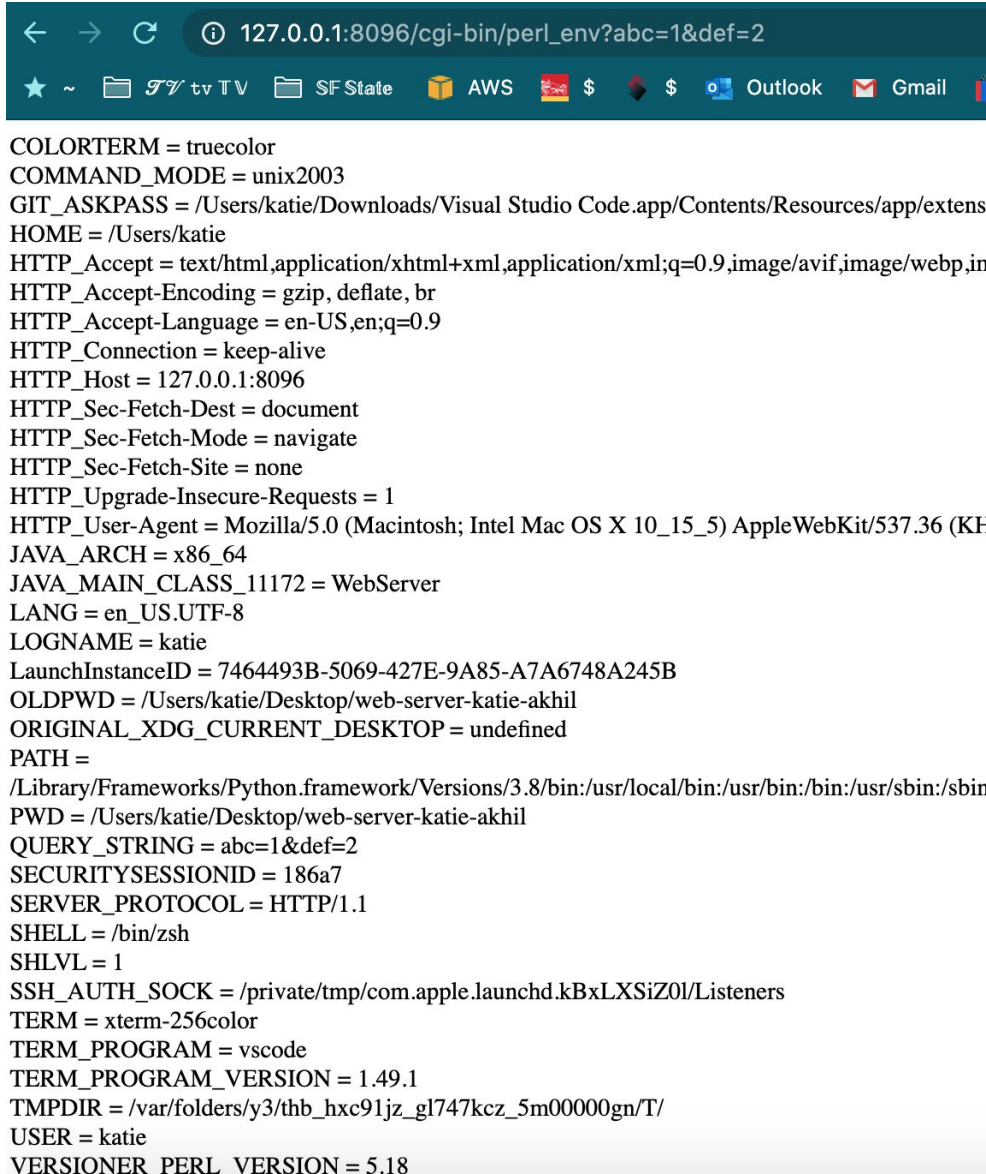
### **What Was Difficult**

One of the most difficult aspects of coding this web server was deciding on the file structure and what class should inherit what object. In different stages of development, we created new class files to handle different aspects of the server as displayed on the flow chart posted on iLearn. Later, we would find out it was easier to instead put more functionality in one class file, so we deleted a few of the files we created.

Another difficulty we had was implementing caching. It took us a while to implement the 304 response trigger because handling the time and time zones took a lot longer than we thought it would. We ended up using a formatting class to make checking the timestamps against each other easier.

## **Testing Process**

We tested each class file's functionality with the file named `Testing.java` and with individual methods exclusively created for testing within each class. The following image shows our web server's response output for a successfully processed request (HTTP code 200):

A screenshot of a web browser window. The address bar shows the URL "127.0.0.1:8096/cgi-bin/perl\_env?abc=1&def=2". The browser's taskbar at the bottom includes icons for various applications like VS Code, SF State, AWS, and Outlook. The main content area of the browser displays a list of environment variables in a monospaced font. The variables include system settings like COLORTERM, COMMAND\_MODE, and HOME, as well as HTTP-related headers like HTTP\_Accept and HTTP\_Host. It also shows user-specific information like LOGNAME and the current directory PATH.

```
COLORTERM = truecolor
COMMAND_MODE = unix2003
GIT_ASKPASS = /Users/katie/Downloads/Visual Studio Code.app/Contents/Resources/app/extens
HOME = /Users/katie
HTTP_Accept = text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,in
HTTP_Accept-Encoding = gzip, deflate, br
HTTP_Accept-Language = en-US,en;q=0.9
HTTP_Connection = keep-alive
HTTP_Host = 127.0.0.1:8096
HTTP_Sec-Fetch-Dest = document
HTTP_Sec-Fetch-Mode = navigate
HTTP_Sec-Fetch-Site = none
HTTP_Upgrade-Insecure-Requests = 1
HTTP_User-Agent = Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36 (KF
JAVA_ARCH = x86_64
JAVA_MAIN_CLASS_11172 = WebServer
LANG = en_US.UTF-8
LOGNAME = katie
LaunchInstanceID = 7464493B-5069-427E-9A85-A7A6748A245B
OLDPWD = /Users/katie/Desktop/web-server-katie-akhil
ORIGINAL_XDG_CURRENT_DESKTOP = undefined
PATH =
/Library/Frameworks/Python.framework/Versions/3.8/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
PWD = /Users/katie/Desktop/web-server-katie-akhil
QUERY_STRING = abc=1&def=2
SECURITYSESSIONID = 186a7
SERVER_PROTOCOL = HTTP/1.1
SHELL = /bin/zsh
SHLVL = 1
SSH_AUTH_SOCK = /private/tmp/com.apple.launchd.kBxLXSiz0l/Listeners
TERM = xterm-256color
TERM_PROGRAM = vscode
TERM_PROGRAM_VERSION = 1.49.1
TMPDIR = /var/folders/y3/thb_hxc91jz_gl747kcz_5m00000gn/T/
USER = katie
VERSIONER_PERL_VERSION = 5.18
```