

Mini-Project 1 Report

Step 1: Status Code Generation Logic

For each status code that the web server needs to generate, we've implemented the logic as follows:

1. 200 OK:

Logic:

- a. The server returns a *200 OK* status when the client makes a valid request for an existing resource. This means the requested file is present in the server's directory, and the server successfully reads and sends the file content to the client.
- b. The response includes the file's content, along with headers such as *Content-Type* and *Last-Modified*.

HTTP Request:

```
GET /test.html HTTP/1.1  
Host: localhost:8080
```

2. 304 Not Modified:

Logic:

- a. The server returns a '304 Not Modified' status when the client makes a conditional GET request with the 'If-Modified-Since' header, and the requested resource has not been modified since the specified date and time.
- b. In this case, the server checks the 'Last-Modified' timestamp of the requested file. If it is earlier than or equal to the date provided in the 'If-Modified-Since' header, the server responds with '304 Not Modified', meaning the client's cached version is still up to date.

HTTP Request:

```
GET /test.html HTTP/1.1  
Host: localhost:8080  
If-Modified-Since: Wed, 23 Oct 2024 07:28:00 GMT
```

3. 400 Bad Request:

Logic:

- a. The server returns a '400 Bad Request' status when the client sends an invalid or malformed request. This can happen if the request line is

improperly formatted, missing essential components, or includes invalid syntax.

- b. The server will not attempt to process the request further and will respond with an error message.

HTTP Request:

GET /test..html HTTP/1.1

Host: localhost:8080

4. 404 Not Found:

Logic:

- a. The server returns a '404 Not Found' status when the client requests a resource that does not exist on the server. This could happen if the file is missing or if the client mistypes the file name.
- b. The server checks its directory for the requested file, and if it cannot be found, it responds with a '404 Not Found' status.

HTTP Request:

GET /nonexistent.html HTTP/1.1

Host: localhost:8080

5. 501 Not Implemented:

Logic:

- a. The server returns a '501 Not Implemented' status when the client sends a request using an HTTP method that the server does not support. In this case, if the server only supports 'GET' requests and the client attempts to send a 'POST' request, the server will respond with '501 Not Implemented'.
- b. The server will not process the request further and will return an error message to the client.

HTTP Request:

POST /test.html HTTP/1.1

Host: localhost:8080

Step 2: Testing the Web Server

(b) Testing Your Web Server with a Basic Request



(c) Testing the Status Codes with curl

To test the web server, we used curl commands from the terminal. The following commands were executed for each status code:

1. **200 OK:**
`'curl -v http://127.0.0.1:8080/test.html'`

```
[prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -v http://localhost:8080/test.html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 62889 failed: Connection refused
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080
> GET /test.html HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

  <p>Congratulations! Your Web Server is Working!</p>

</body>

</html>
* Connection #0 to host localhost left intact
```

2. 304 Not Modified:

'curl -v -H "If-Modified-Since: Wed, 23 Oct 2024 07:28:00 GMT"
http://127.0.0.1:8080/test.html'

```
[prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -v -H "If-Modified-Since: Wed, 23 Oct 2024 07:28:00 GMT" http://localhost:8080/test.html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 62925 failed: Connection refused
* Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080
> GET /test.html HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
> If-Modified-Since: Wed, 23 Oct 2024 07:28:00 GMT
>
* Request completely sent off
< HTTP/1.1 304 Not Modified
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
* Connection #0 to host localhost left intact
```

3. 400 Bad Request:

'curl -v <http://127.0.0.1:8080/test..html>'

```
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -v http://localhost:8080/test..html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 62951 failed: Connection refused
* Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080
> GET /test..html HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 400 Bad Request
< Content-Type: text/html
< Connection: close
<
* Closing connection
<h1>400 Bad Request</h1>>
```

4. **404 Not Found:**

'curl -v <http://127.0.0.1:8080/nonexistent.html>'

```
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -v http://localhost:8080/nonexistent.html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 62973 failed: Connection refused
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080
> GET /nonexistent.html HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 404 Not Found
< Content-Type: text/html
< Connection: close
<
* Closing connection
<h1>404 Not Found</h1>
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 %
```

5. **501 Not Implemented:**

'curl -v -X POST <http://127.0.0.1:8080/test.html>'

```
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -v -X POST http://localhost:8080/test.html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* connect to ::1 port 8080 from ::1 port 62987 failed: Connection refused
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080
> POST /test.html HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 501 Not Implemented
< Content-Type: text/html
< Connection: close
<
* Closing connection
<h1>501 Not Implemented</h1><p>The method POST is not supported.</p>
```

Step 3: Performance

(a) Specifications for Proxy Server

Proxy server requirement:

1. Receives HTTP requests from clients intended for external web servers.
2. Parses the request to determine the target server.
3. Establishes a connection to the target server, forwards the request, and relays the response back to the client.

4. Caching: Can cache frequently requested resources to improve performance and reduce bandwidth usage.

(b) Test Procedure for Proxy Server

To verify the proxy server's functionality, I used curl commands similar to the ones in Step 2 but passed through the proxy server:

1. **200 OK Test via Proxy:**
`'curl -x localhost:8888 http://127.0.0.1:8080/test.html -v'`
2. **304 Not Modified Test via Proxy:**
`'curl -x localhost:8888 -v -H "If-Modified-Since: Wed, 23 Oct 2024 07:28:00 GMT" http://127.0.0.1:8080/test.html'`
3. **400 Bad Request Test via Proxy:**
`'curl -x localhost:8888 http://127.0.0.1:8080/test..html -v --raw'`
4. **404 Not Found Test via Proxy:**
`'curl -x localhost:8888 http://127.0.0.1:8080/nonexistent.html -v'`
5. **501 Not Implemented Test via Proxy:**
`'curl -x localhost:8888 -X POST http://127.0.0.1:8080/test.html -v'`

(c) Multi-threading in Web Server

Our web server is a multithreaded server, the reason our server is multi-thread is every connection we create is on a new thread. The multi-thread server can enhance throughput, simplify the programming model, and efficiently utilize resources.

Screenshots:

```
[prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -x localhost:8888 http://127.0.0.1:8080/test.html -v
* Host localhost:8888 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8888...
* connect to ::1 port 8888 from ::1 port 63145 failed: Connection refused
* Trying 127.0.0.1:8888...
* Connected to localhost (127.0.0.1) port 8888
> GET http://127.0.0.1:8080/test.html HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/8.7.1
> Accept: */*
> Proxy-Connection: Keep-Alive
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

  <p>Congratulations! Your Web Server is Working!</p>

</body>

</html>
* Connection #0 to host localhost left intact
```



```
[prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl -x localhost:8888 http://127.0.0.1:8080/test.html -v
* Host localhost:8888 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8888...
* connect to ::1 port 8888 from ::1 port 63194 failed: Connection refused
* Trying 127.0.0.1:8888...
* Connected to localhost (127.0.0.1) port 8888
> GET http://127.0.0.1:8080/test.html HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/8.7.1
> Accept: */*
> Proxy-Connection: Keep-Alive
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

  <p>Congratulations! Your Web Server is Working!</p>

</body>

</html>
* Connection #0 to host localhost left intact
```

```
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % curl http://127.0.0.1:8080/test.html -v &
curl http://127.0.0.1:8080/test.html -v &
[1] 42869
[2] 42870
[3] 42871
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 % * * Trying 127.0.0.1:8080...
    Trying 127.0.0.1:8080...
*   Trying 127.0.0.1:8080...
* Connected to 127.0.0.1 (127.0.0.1) port 8080
* Connected to 127.0.0.1 (127.0.0.1) port 8080
> GET /test.html HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* > * GET /test.html HTTP/1.1
Connected to 127.0.0.1 (127.0.0.1) port 8080
> Host: 127.0.0.1:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
> GET /test.html HTTP/1.1
Request completely sent off
> Host: 127.0.0.1:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

<   <p>Congratulations! Your Web Server is Working!</p>
HTTP/1.1 200 OK

</body>

< </html>
Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>
```

```
HTTP/1.1 200 OK

</body>

< </html>
Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working!</p>

</body>

</html>
< HTTP/1.1 200 OK
< Content-Type: text/html
< Transfer-Encoding: chunked
< Last-Modified: Tue, 22 Oct 2024 01:20:53 GMT
< Connection: keep-alive
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working!</p>

</body>

</html>
* Connection #0 to host 127.0.0.1 left intact
* Connection #0 to host 127.0.0.1 left intact
* Connection #0 to host 127.0.0.1 left intact

[2] - done      curl http://127.0.0.1:8080/test.html -v
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 %
[3] + done      curl http://127.0.0.1:8080/test.html -v
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 %
[1] + done      curl http://127.0.0.1:8080/test.html -v
prateeksingh@Prateeks-MacBook-Pro-M1-3 cmpt-371-mp1 %
```

Step Four: Expand

To avoid the Head-of-Line (HOL) blocking problem, we implemented HTTP/2 features such as interleaving frames to ensure that multiple requests can be processed in parallel without being blocked by a single resource.

- **Explanation:** In HTTP/2, requests and responses are split into small frames and sent interleaved over a single connection. This avoids the blocking problem that occurs when one large response holds up the rest.
- **Implementation Changes:** The server now chunks responses and sends them in smaller pieces, allowing it to serve parts of different responses simultaneously.