**DV201 (Software Engineering ) Assignment 2 ( HTTP Server )**

**By: Alex and Fabian**

Problem 1:

Named HTML Page: This test was preformed by typing in the URL: “localhost:8080/fun.html” into the browser and taking a screenshot of the response. It retrieves the file fun.html from the root of the server.

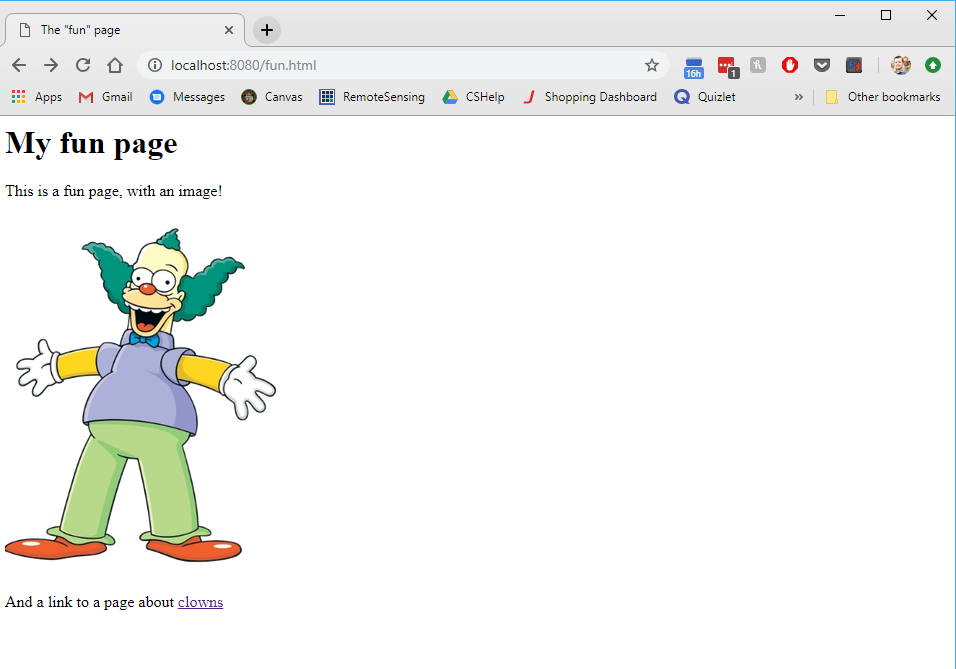
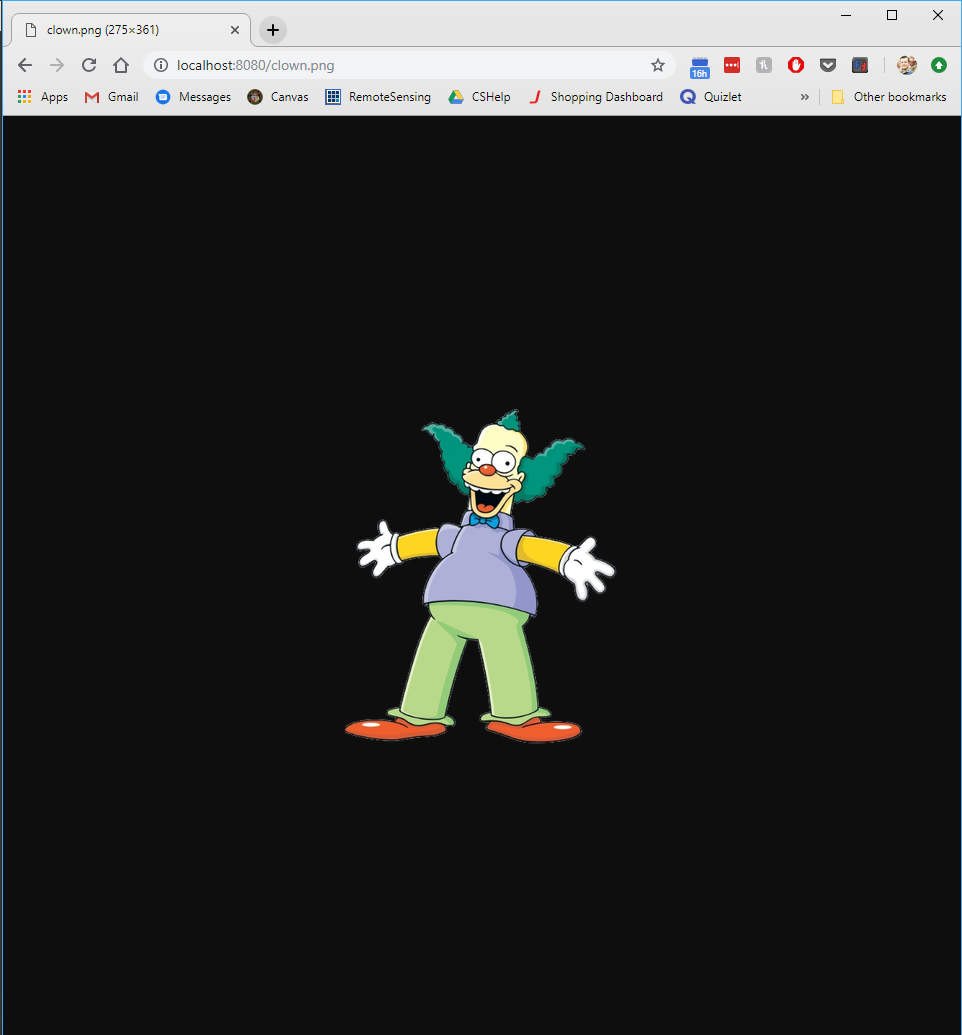
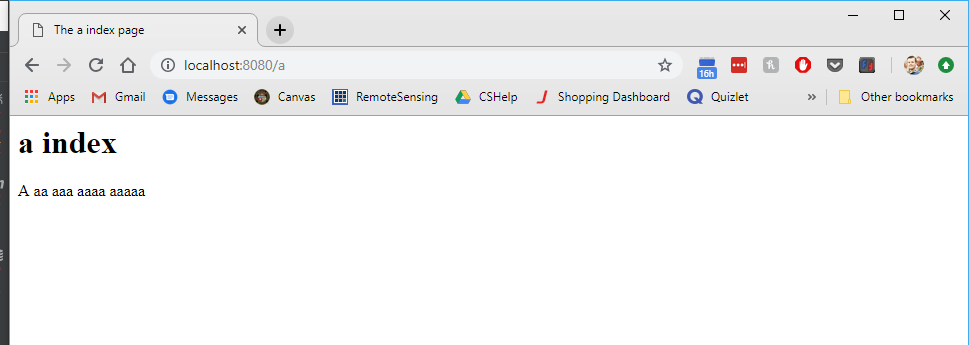


Image: This test retrieves a PNG image from the server. It was preformed by entering the URL: “localhost:8080/clown.png” where clown.png is a PNG image on the root of the server.

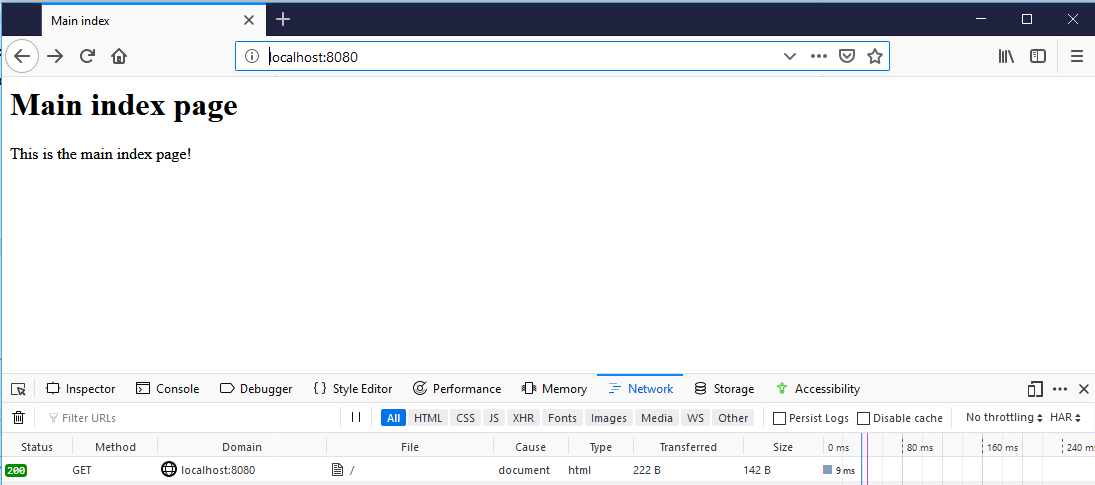
****

A directory with index.html file: This test was preformed by entering “localhost:8080/a” a brower. It returns the index.html file present in the a directory on the server.

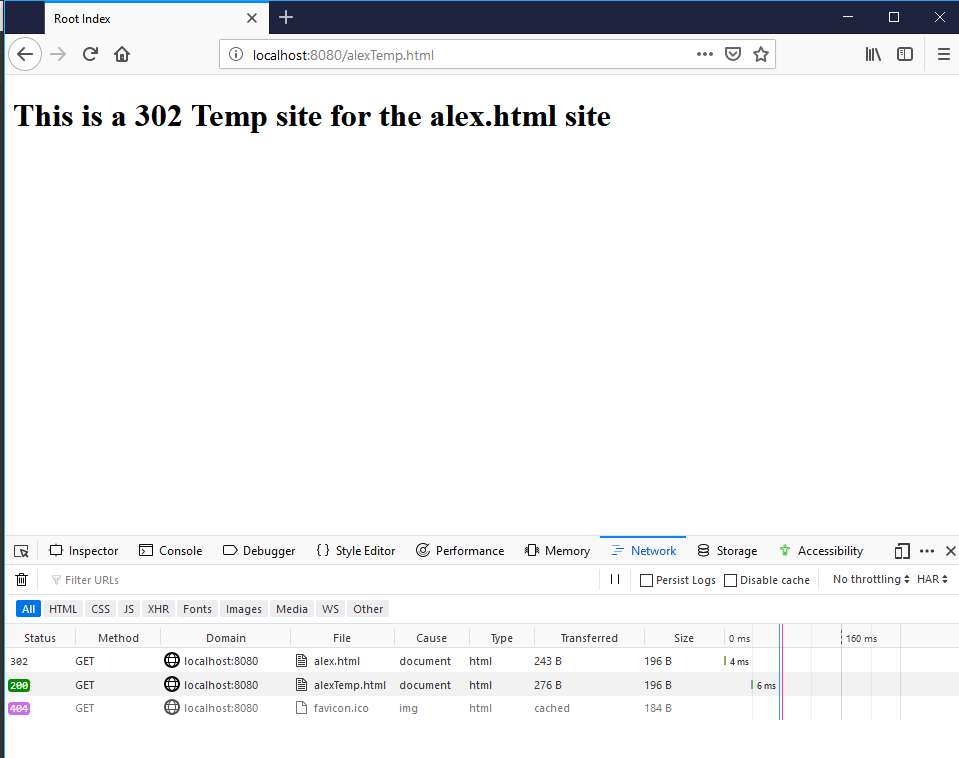


**Problem 2:**

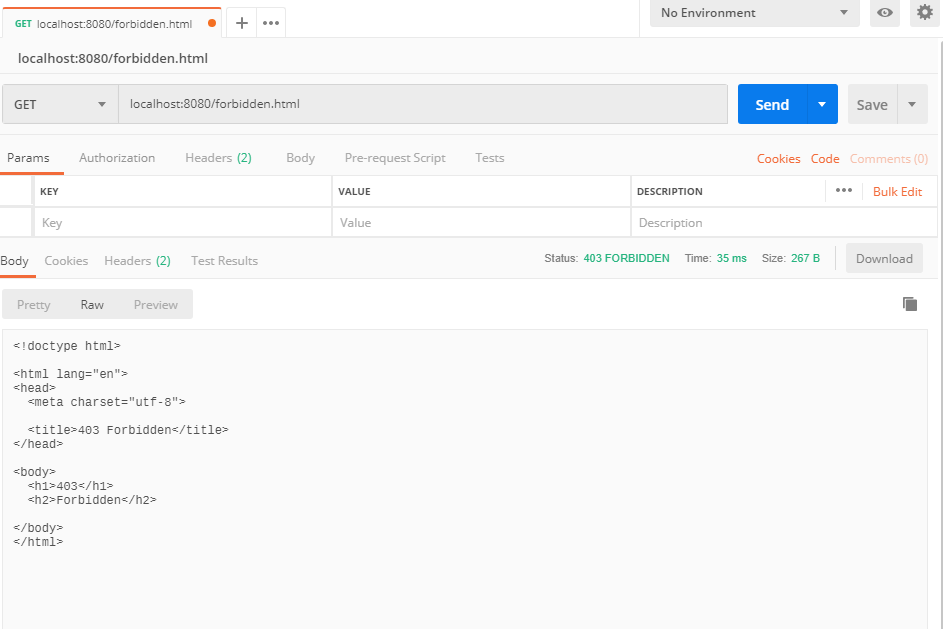
**200 OK:** For a GET request, the 200 OK response is returned when the requested resource exists and is available (not forbidden or moved) to the user. This test was preformed by entering: “localhost:8080” into a browser and it returns the index.html page present at the root of the server. As displayed in the browser the index.html page was returned. As displayed in the bottom section of the screenshot the GET request issued to the server returned a 200 OK response.



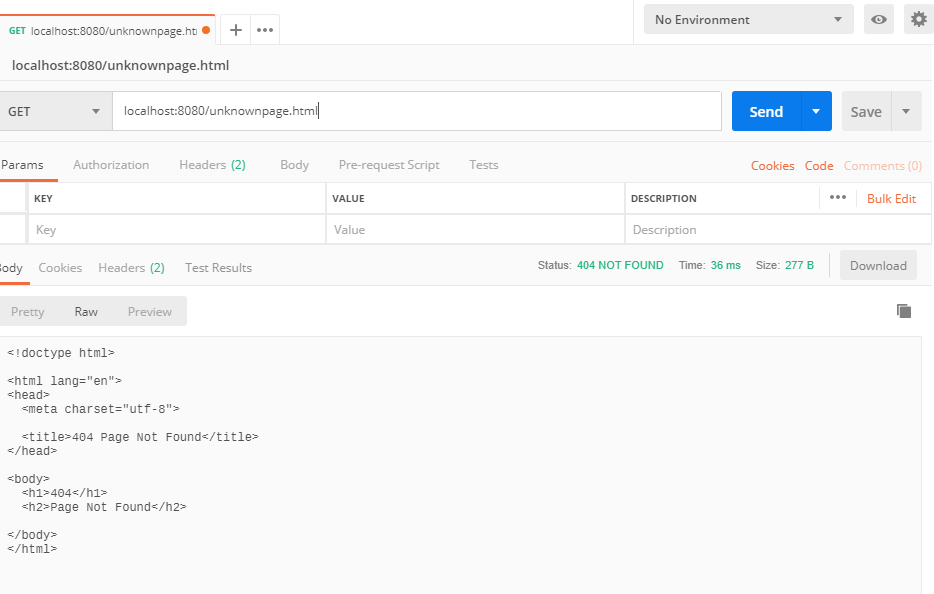
**302 Found:** For a GET request, the 302 Found code indicates that the requested resource is temporarily located under a different URI. On our server we have a Map that contains key value pairs. The key is the requested resource URI and the value is the new temporary resource URI. When the server receives a GET request it checks the map to see if the request resource matches a key in the redirect map. If it does it returns a 302 Response as shown in the screenshot below when “localhost:8080/alex.html” was requested. Then the server provides the new temporary resource (in this case alexTemp.html) and returns a 200 OK status afterwards to indicate that the resource is available and ok.



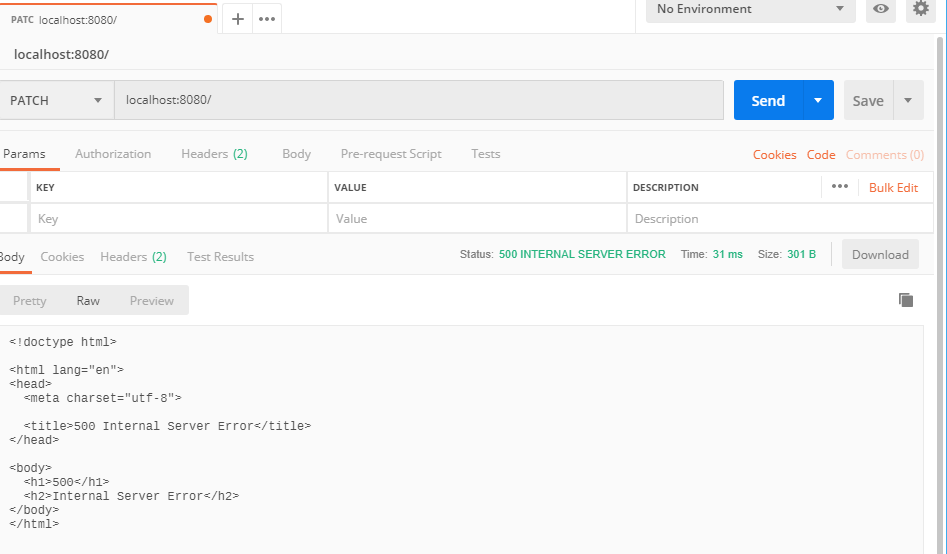
**403 Forbidden:** For a GET request, the 403 Forbidden response code indicates that the server understands the request, but the resource is not allowed to be accessed. Authentication will not help and there is nothing that the client can do to access the resource so they should not reissue the request. Our server contains a list of resources that are forbidden and it includes “localhost:8080/forbidden.html” This test was conducted by requesting “localhost:8080/forbidden.html” from the POSTman API testing tool (Postman). The server returned a 403 FORBIDDEN response as indicated by the Status section of the screenshot and it displayed the 403.html resource from the server to show the client the resource is forbidden.



**404 Not Found:** The 404 Not Found response indicates that the requested URI cannot be found on the server. This test was conducted by requesting “unknownpage.html” from the server. Since “unknownpage.html” does not exist on the server it returns a 404 NOT FOUND response as shown in the Status: section of the screenshot below and it serves the 404.html resource from the server to show the client the page cannot be found.

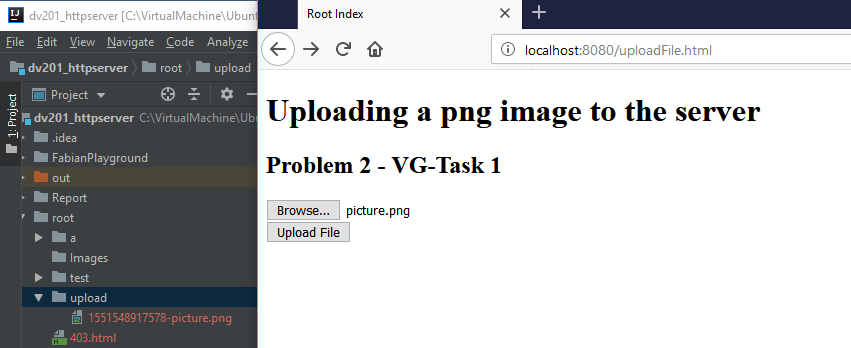


**500 Internal Error:** The 500 internal server error response indicates that the server ran into an unexpected condition that prevented it from fulfilling the request. There are many ways to generate a 500 response both expected and unexpected. In this test we requested an unsupported HTTP request type as shown in the screenshot below. We issued a PATCH request to the server and since our server does not support PATCH requests it returned a 500 Internal Server Error. The server served the 500.html resource to show the client that there was an internal server error.

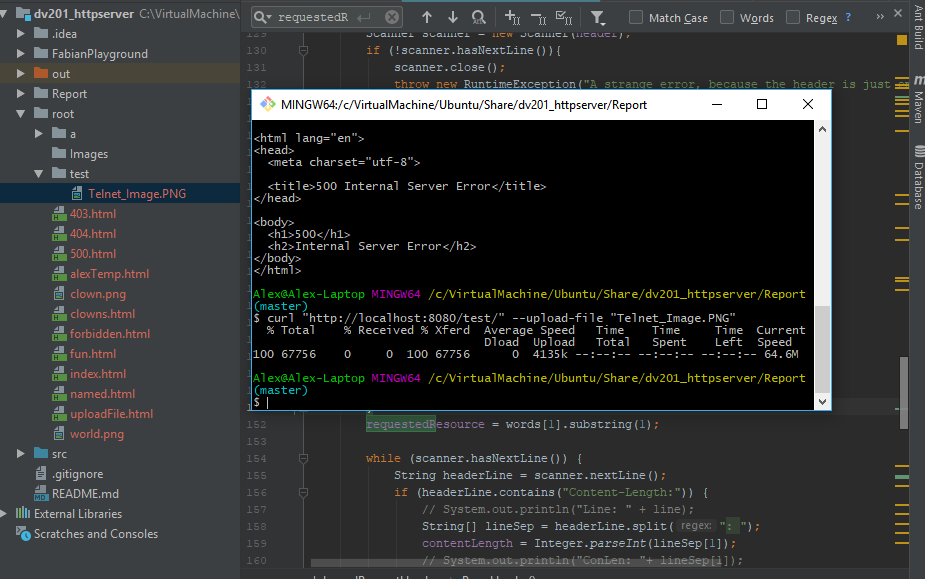


VG Task 1:

For this task we have an html form that uses the enc-type “multipart/form-data.” This allows us to submit the filename and the binary file data to the server in the body of the POST request. The test was performed by accessing the “uploadFile.html” page and selecting a PNG image from my file system. Then the Upload File button was pressed. When the server receives the POST request it will always create a new resource in the upload folder on the server. Regardless of the filename there will always be a new resource because a date stamp is appended to each file name. The user is only able to upload a file; the server handles the creation of the resource and where it will be placed on the server.



VG Task 2:



PUT vs. POST:

A POST request should be used to modify and update a resource. A PUT request should be used to create a resource or overwrite an existing one. A main difference between the two is that POST should not be used to create resources and if a resource does not exist it will give an error. With a PUT request you specify the resources new URL and the request is idempotent. This means that regardless of how many times you execute the request and regardless of weather it previously existed the result form the request will be the same. A POST request is not idempotent. A post request can update or create a child resource at a predefined URL on the server. For example, if you generate a POST request to /items then a new resource will be created on the server that lives under /items for example /items/1. If you send the request multiple times, multiple resources will be generated. PUT updates or creates resources by replacing them in their entirety.

**Problem 3 (Telnet):**

Named HTML Page:

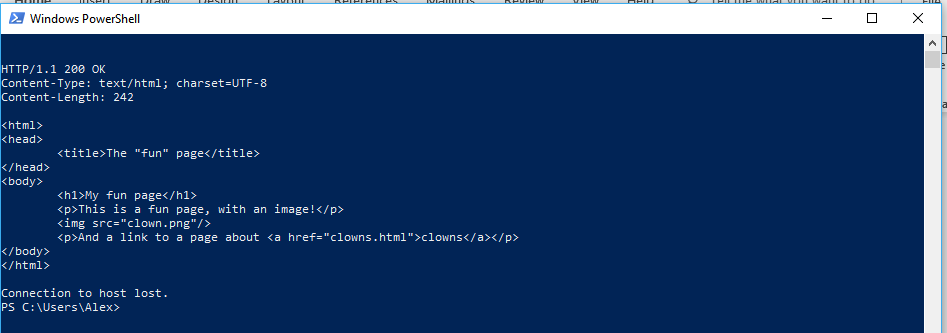
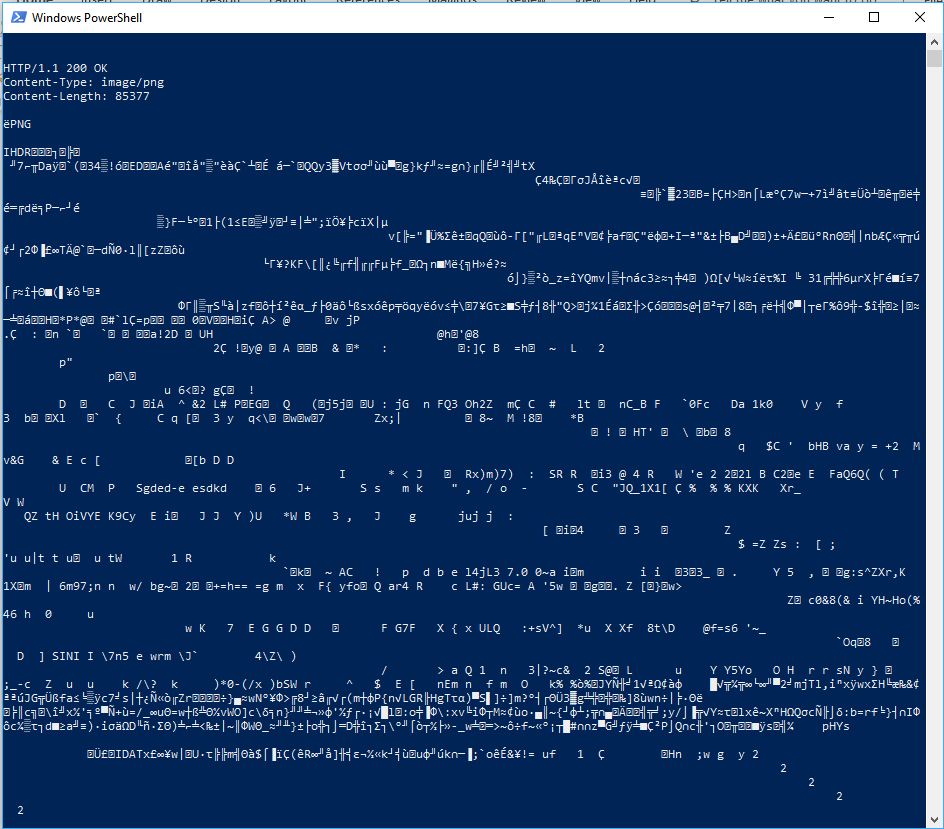


Image:  


Directory:

