Linnéuniversitetet
Kalmar Växjö

Report

Assignment 2

1DV701

Authors: Ibrahim Groshar & Nedko Nedkov
Semester: Spring 2023
Email:
ig222je@student.lnu.se
nn222mx@student.lnu.se

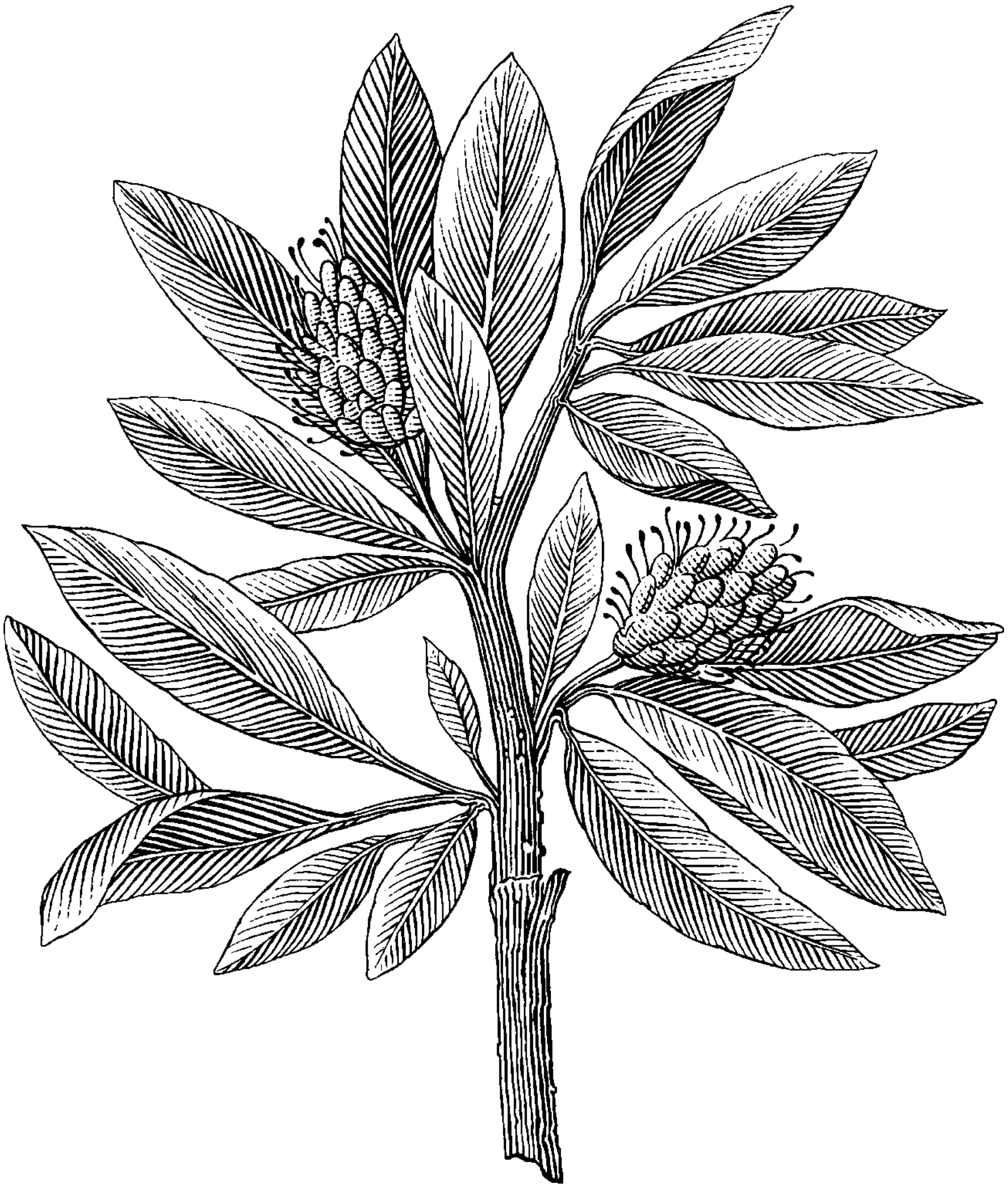
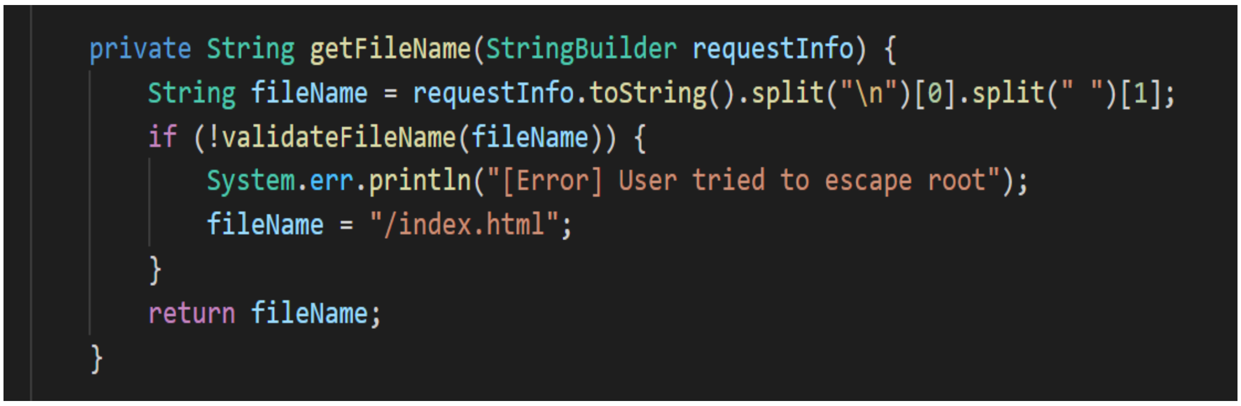


Table of Contents

1 Discussion

The provided code implements a working HTTP server with support for HTML, HTM, and image files. The server is designed to run on port 80 by default. The server also includes safety measures to prevent clients from escaping the web root directory and accessing confidential information on the server.
The try-catch blocks throughout the code help to ensure that any exceptions that appear are handled, preventing the server from crashing. All resources are closed after use to prevent leaks. There are 4 HTTP status codes implemented in the program, which are used to indicate status of client requests. The ServerConn class includes methods that send status codes for 200 (OK), 302 (Found), 404 (Not Found), and 500 (Internal Server Error) responses, depending on the outcome of the request.

To ensure the safety of the server, the code includes measures to prevent clients from escaping the web root directory and accessing sensitive files. Specifically, the validateFileName method in the ServerConn class checks that the full path starts with the given root. If an invalid root directory is provided, the code removes the escape characters in the directory. The ServerConn class also includes a method to check that requested filenames are complete and match a specific pattern to prevent clients from requesting files outside of the web root directory.
Escaping the root also results in redirection to index.html.

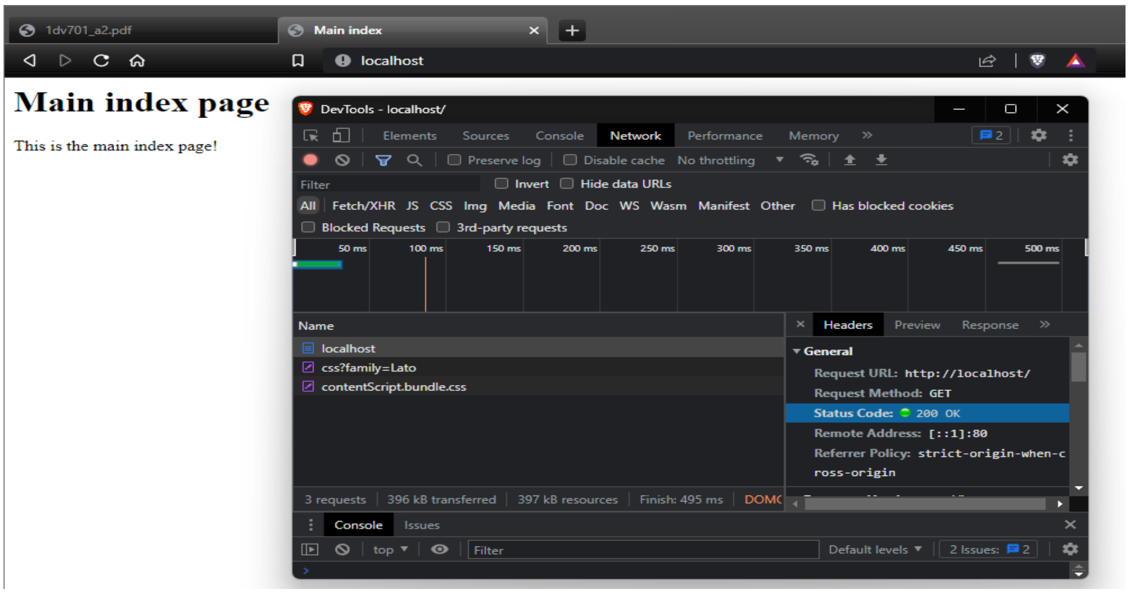

```java
private String getFileName(StringBuilder requestInfo) {
    String fileName = requestInfo.toString().split("\n")[0].split(" ")[1];
    if (!validateFileName(fileName)) {
        System.err.println("[Error] User tried to escape root");
        fileName = "/index.html";
    }
    return fileName;
}
```

Finally, the code includes console output to provide information about the status of the server and client connections.

When a client connects to the server, it’s data like IP address, port and other connection details are outputted by the Server class. The ServerConn class outputs information about the HTTP request headers it receives from the client. The code is built on a robust and secure HTTP server that can handle multiple client request via multithreading. By including exception handling, HTTP status code implementations, and safety measures to prevent clients from accessing sensitive files, the code demonstrates best practices for server-side programming.

2 HTTP Response Codes

2.1 Status code 200



Main index page

This is the main index page!

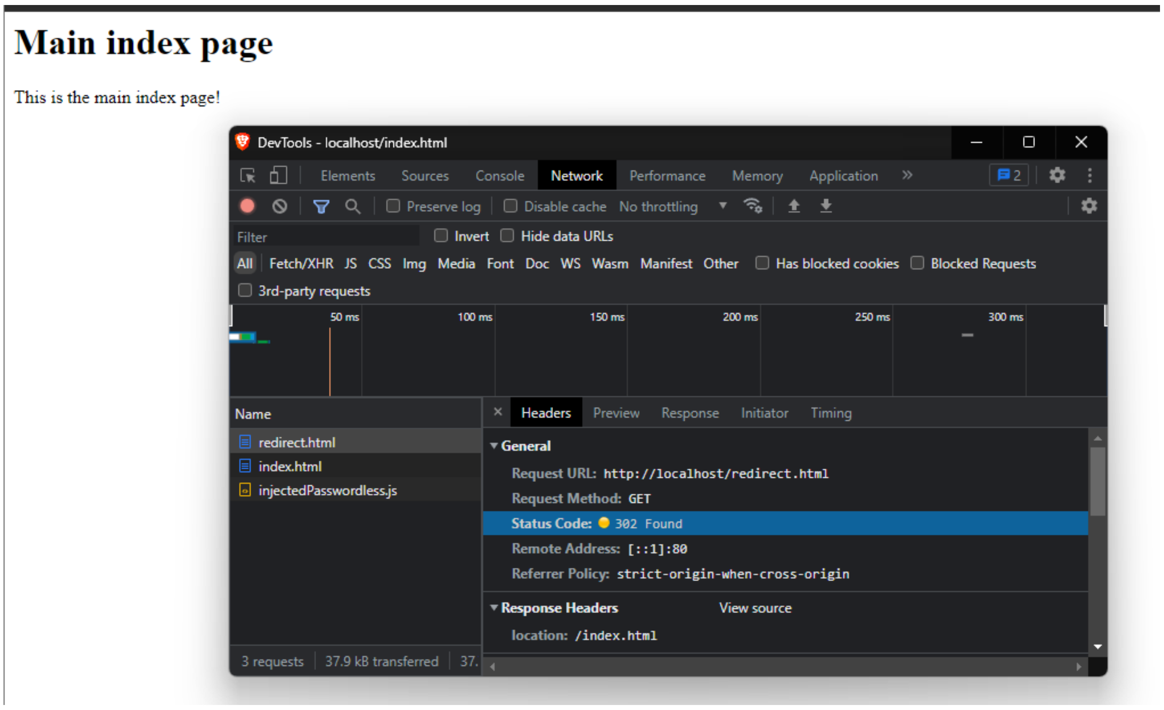
Name
localhost
css?family=Lato
contentScript.bundle.css

General
Request URL: http://localhost/
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin

3 requests | 396 kB transferred | 397 kB resources | Finish: 495 ms | DOMC

This status code indicates that the client's request has been successfully processed and the server returns the requested resource to the client.

2.2 Status code 302



Main index page

This is the main index page!

Name
redirect.html
index.html
injectedPasswordless.js

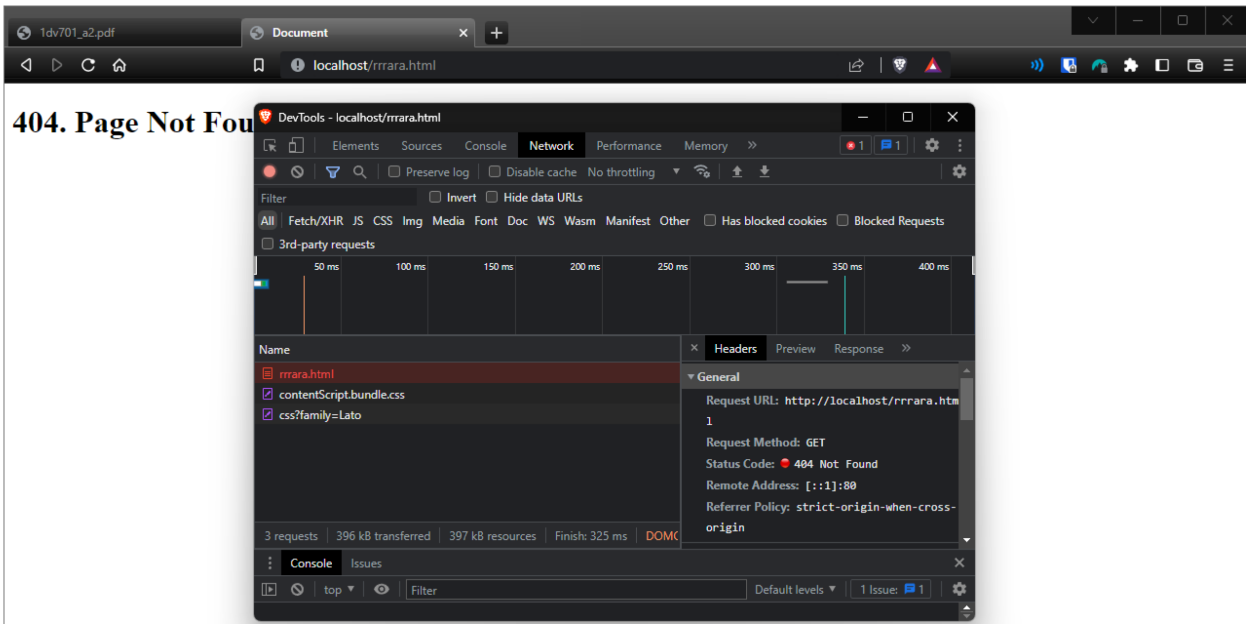
General
Request URL: http://localhost/redirect.html
Request Method: GET
Status Code: 302 Found
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin

Response Headers
location: /index.html

3 requests | 37.9 kB transferred | 37.

This is a redirect response that indicates when a requested resource has temporarily moved to the URL specified by the header. In our instance when status code 302 is triggered, the client is redirected to the index.html page.

2.3 Status code 404



localhost/rrara.html

404. Page Not Fou

DevTools - localhost/rrara.html

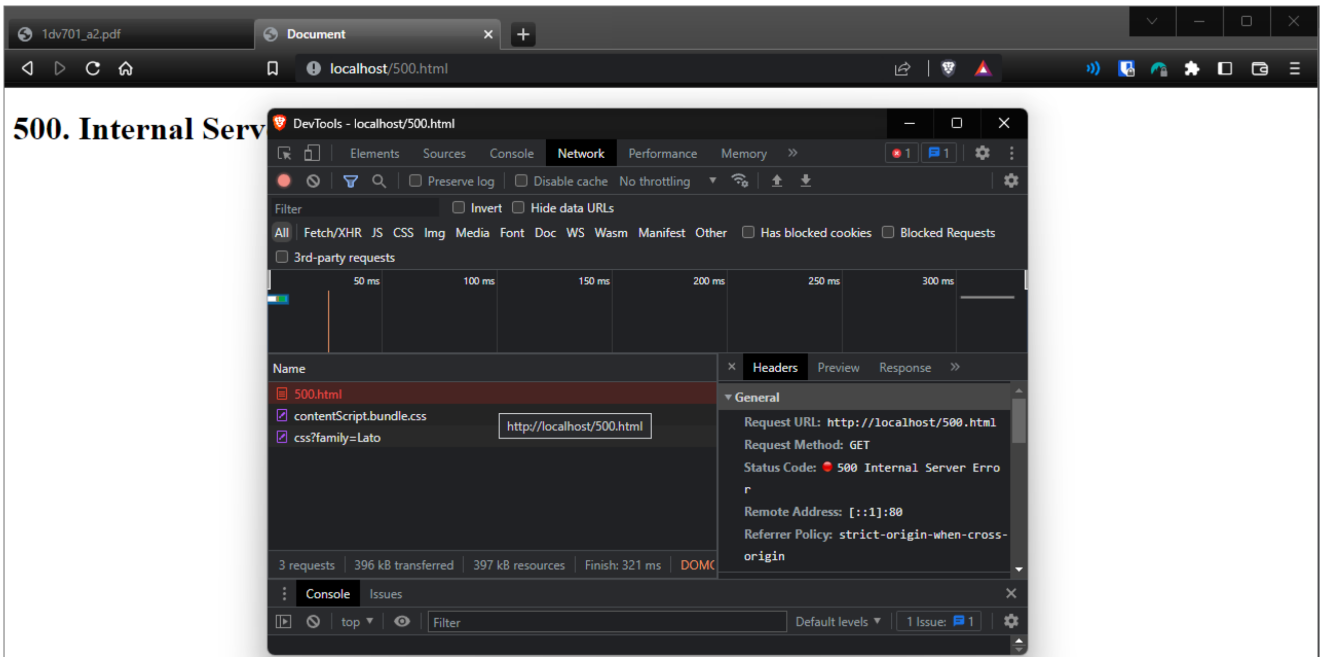
Name
rrara.html
contentScript.bundle.css
css?family=Lato

3 requests | 396 kB transferred | 397 kB resources | Finish: 325 ms | DOMC

```
▾ General
Request URL: http://localhost/rrara.html
Request Method: GET
Status Code: ● 404 Not Found
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin
```

This response status code indicates that the server cannot find the requested resource on the server.

2.4 Status code 500



localhost/500.html

500. Internal Serv

DevTools - localhost/500.html

Name
500.html
contentScript.bundle.css
css?family=Lato

http://localhost/500.html

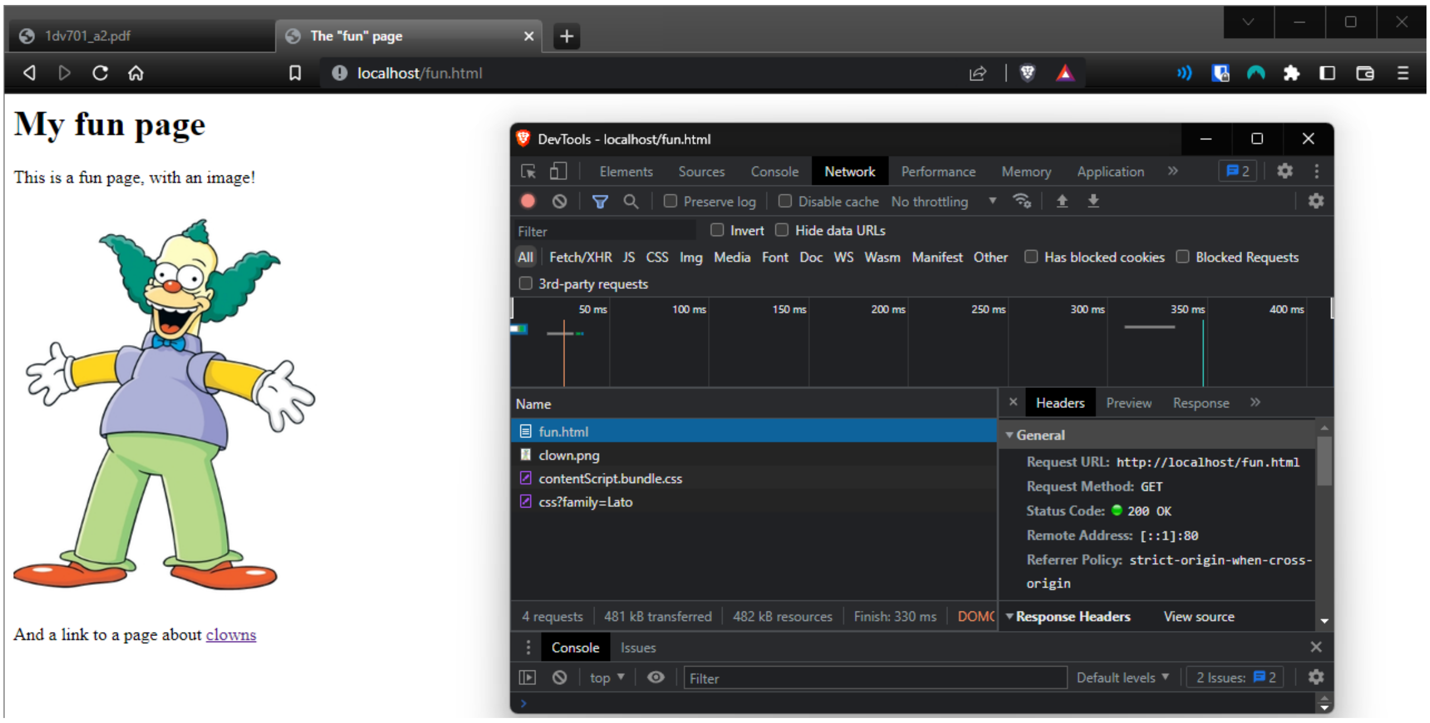
3 requests | 396 kB transferred | 397 kB resources | Finish: 321 ms | DOMC

```
▾ General
Request URL: http://localhost/500.html
Request Method: GET
Status Code: ● 500 Internal Server Error
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin
```

This is an internal server error. It mostly occurs when there was an error on the server while processing the client’s request due to misconfiguration or a problem with the server software.

3 Request Tests

3.1 Requesting HTML page



My fun page

This is a fun page, with an image!

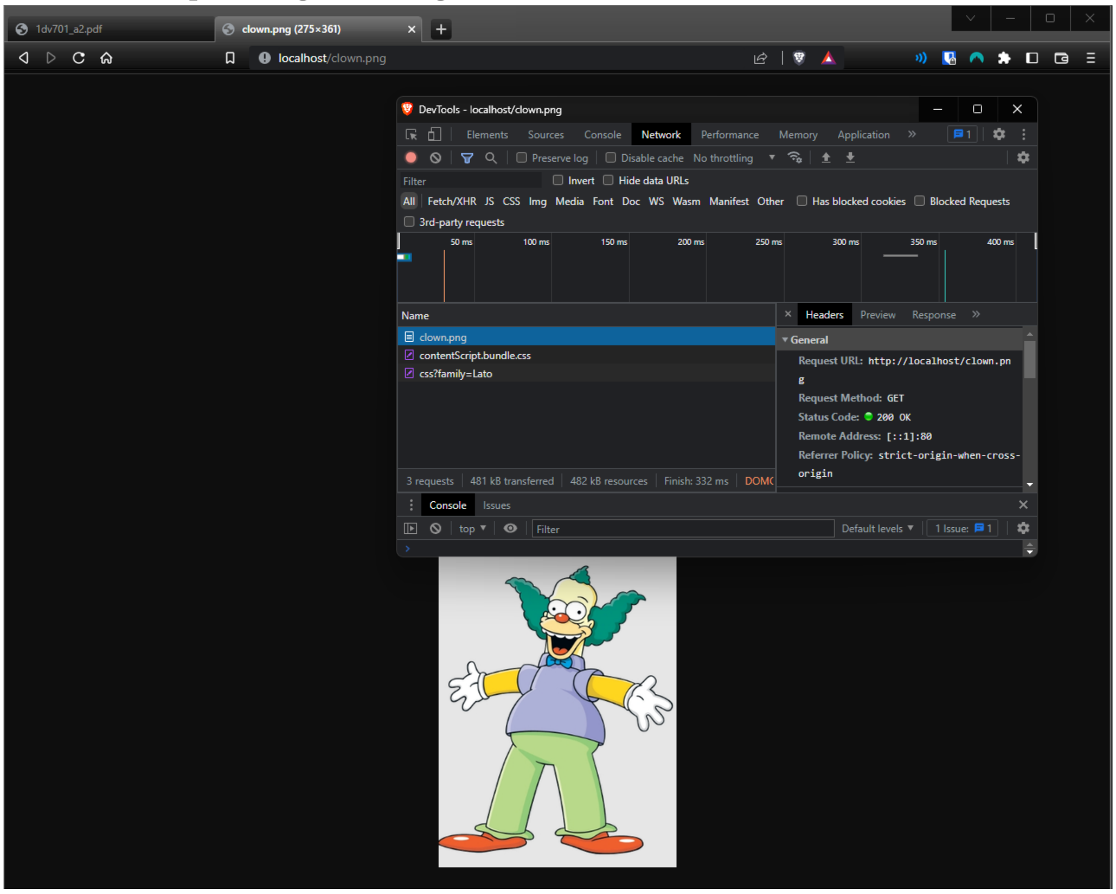
And a link to a page about clowns

```
Name
fun.html
clown.png
contentScript.bundle.css
css?family=Lato

4 requests | 481 kB transferred | 482 kB resources | Finish: 330 ms | DOMC

▾ General
  Request URL: http://localhost/fun.html
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: [::1]:80
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers        View source
```

3.2 Requesting an image

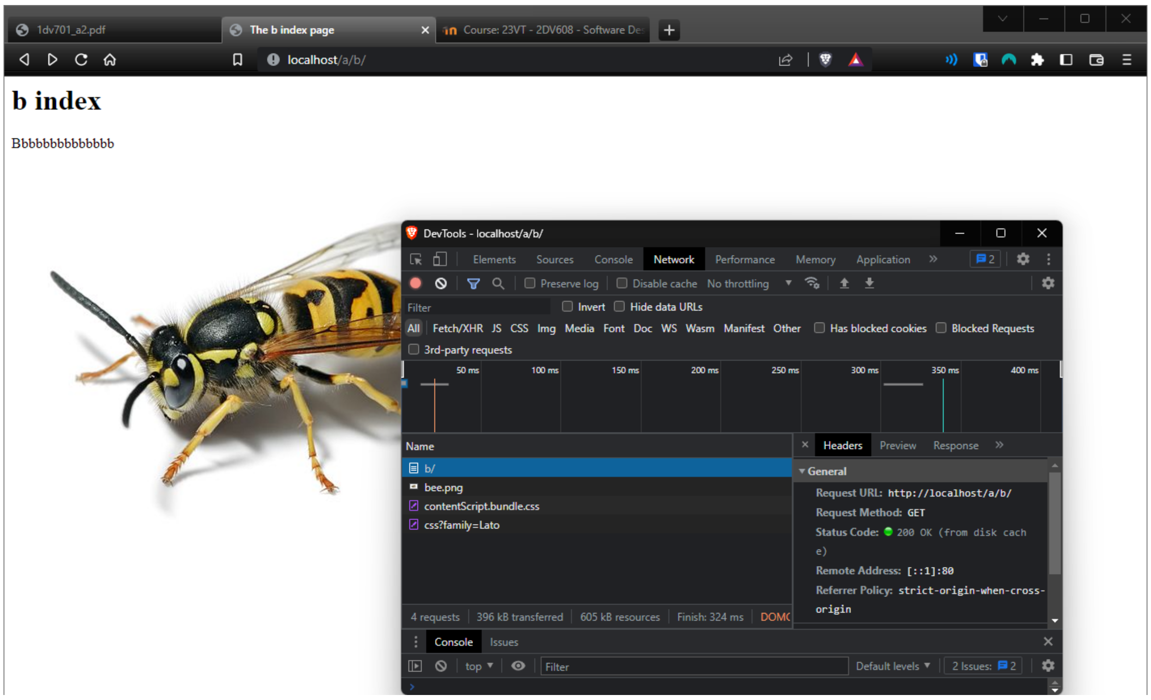


```
Name
clown.png
contentScript.bundle.css
css?family=Lato

3 requests | 481 kB transferred | 482 kB resources | Finish: 332 ms | DOMC

▾ General
  Request URL: http://localhost/clown.png
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: [::1]:80
  Referrer Policy: strict-origin-when-cross-origin
```

3.3 Requesting a directory



b index

Bbbbbbbbbbbbbb

Name
b/
bee.png
contentScript.bundle.css
css?family=Lato

4 requests | 396 kB transferred | 605 kB resources | Finish: 324 ms

General
Request URL: http://localhost/a/b/
Request Method: GET
Status Code: 200 OK (from disk cache)
Remote Address: [::1]:80
Referrer Policy: strict-origin-when-cross-origin

4 Error Handling

We handle exceptions using the try-catch.
In the Server class, we handle 2 possible exceptions:

- **IOException** when the port is being used by another process, printing a message: [Error] Socket error
- **Exception** when invalid arguments are given, printing a message: [Error] Invalid program arguments. Usage "java Server [port] [dir]". Using defaults

In the ServerConn class, we handle the following exceptions:

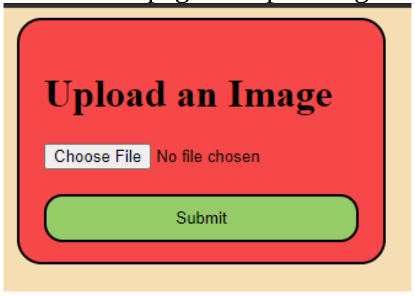
1. **IOException** when buffReader cannot read in run(), printing a message: [Error] Something went wrong. Closing thread
2. **IOException** when the canonical path of a file cannot be taken in validateFileName(), returns that file name is not valid
3. **IOException** when a new file cannot be created in postResponse(), printing a message: [Error] Could not create a file output stream
4. **Exception** when a resource cannot be loaded in getResponse(), returning status code 500 with a custom page
5. **FileNotFoundException** when a file with given name does not exist in getContentLength() and status200(), printing a message: [Error] File does not exist
6. **IOException** when file cannot be read in getContentLength(), printing a message: [Error] Could not read from file
7. **IOException** when status code 500 cannot be shown with its custom page in status500(), printing a message: [Error] Could not show Status 500 Page

5 VG Task

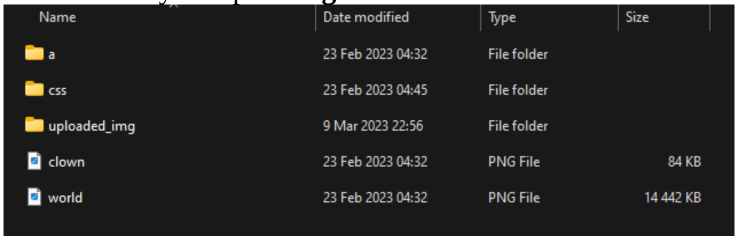
Disclaimer: We tried to implement it but did not succeed.

To post a file, the POST method is used with a file given in a form with multipart/form-data enctype, containing an input with type file, accepting any image is used. The request is then split in 2 parts – header and content. The file bytes are supposed to be in the content but we did not manage to read them properly. The bytes are then put in a byte array, which is written to a FileOutputStream, creating the file. Then our code should redirect the user to the image they uploaded.

The HTML page for uploading is **/upload.html**

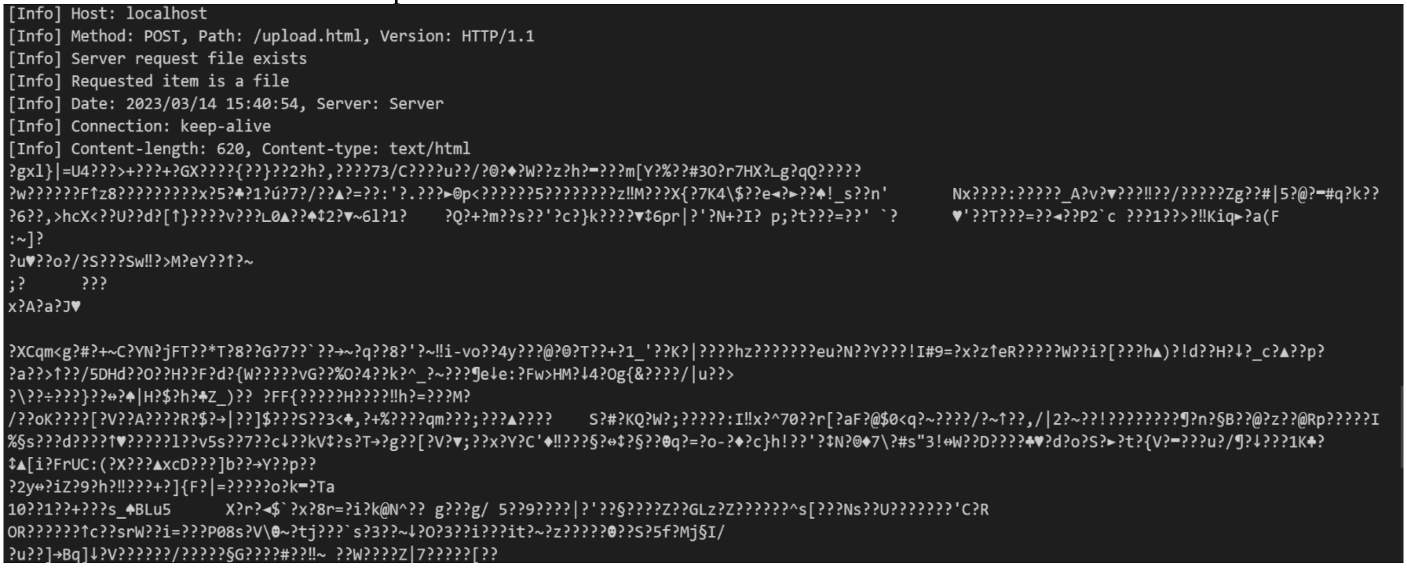


The form only accepts **images**



Name	Date modified	Type	Size
a	23 Feb 2023 04:32	File folder	
css	23 Feb 2023 04:45	File folder	
uploaded_img	9 Mar 2023 22:56	File folder	
clown	23 Feb 2023 04:32	PNG File	84 KB
world	23 Feb 2023 04:32	PNG File	14 442 KB

HTTP VG POST Console Output



```
[Info] Host: localhost
[Info] Method: POST, Path: /upload.html, Version: HTTP/1.1
[Info] Server request file exists
[Info] Requested item is a file
[Info] Date: 2023/03/14 15:40:54, Server: Server
[Info] Connection: keep-alive
[Info] Content-length: 620, Content-type: text/html
?gxl}|=U4???>+???+?GX????{??}??2?h?,????73/C????u??/?☺?♦?W??z?h?━???m[Y?%??#3O?r7HX?ʟg?qQ?????
?w??????F↑z8?????????x?5?♣?1?ú?7?/??▲?=??:'?.???►☺p<??????5????????z‼M???X{?7K4\$??e◄?►??♠!_s??n'      Nx????:?????_A?v?▼???‼??/?????Zg??#|5?@?━#q?k??
?6??,>hcX<??U??d?[↑}????v???ʟ0▲??♠↕2?▼~6l?1?    ?Q?+?m??s??'?c?}k????▼↕6pr|?'?N+?I? p;?t???=??' `?      ♥'??T???=??◄??P2`c ???1??>?‼Kiq►?a(F
:~]?
?u♥??o?/?S???Sw‼?>M?eY??↑?~
;?      ???
x?A?a?J♥

?XCqm<g?#?+~C?YN?jFT??*T?8??G?7??`??→~?q??8?'?~‼i-vo??4y???@?☺?T??+?1_'??K?|????hz???????eu?N??Y???!I#9=?x?z↑eR?????W??i?[???h▲)?!d??H?↓?_c?▲??p?
?a??>↑??/5DHd??O??H??F?d?{W?????vG??%O?4??k?^_?~???¶e↓e:?Fw>HM?↓4?Og{&????/|u??>
?\??÷???}??↔?♠|H?$?h?♣Z_)?? ?FF{?????H????‼h?=???M?
/??oK????[?V??A????R?$?→|??]$???S??3<♣,?+%????qm???;???▲????    S?#?KQ?W?;?????:I‼x?^70??r[?aF?@$0<q?~????/?~↑??,/|2?~??!????????¶?n?§B??@?z??@Rp?????I
%§s???d????↑♥?????1??v5s??7??c↓??kV↕?s?T→?g??[?V?▼;??x?Y?C'♦‼???§?↔↕?§??☻q?=?o-?♦?c}h!??'?↕N?☺♦7\?#s"3!↔W??D????♣♥?d?o?S?►?t?{V?━???u?/¶?↓???1K♣?
↕▲[i?FrUC:(?X???▲xcD???]b??→Y??p??
?2y↔?iZ?9?h?‼???+?]{F?|=?????o?k━?Ta
10??1??+???s_♠BLu5      X?r?◄$`?x?8r=?i?k@N^?? g???g/ 5??9????|?'??§????Z??GLz?Z??????^s[???Ns??U???????'C?R
OR??????↑c??srW??i=???P08s?V\☻~?tj???`s?3??~↓?O?3??i???it?~?z?????☻??S?5f?Mj§I/
?u??]→Bq]↓?V??????/?????§G????#??‼~ ??W????Z|7?????[??
```