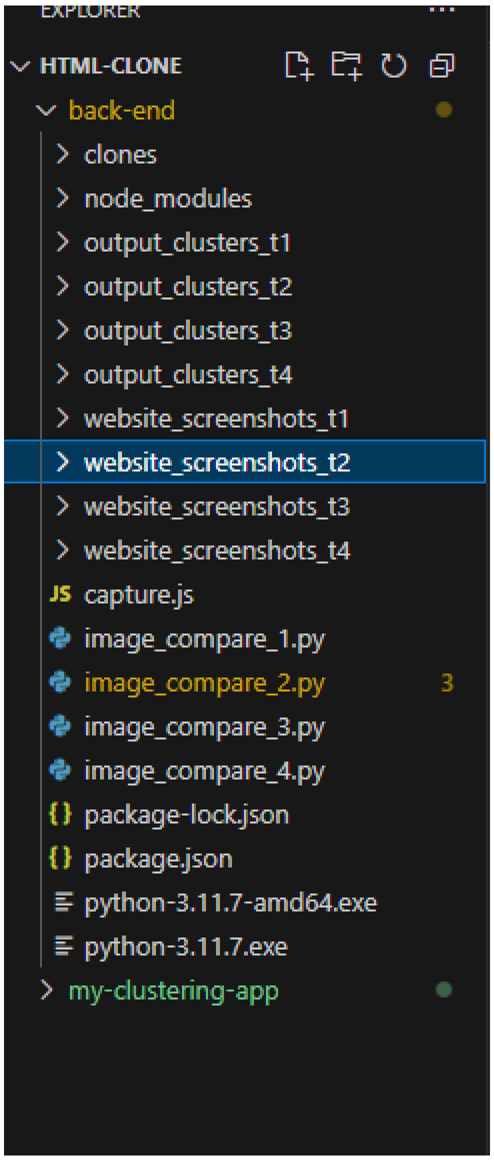
Why did I choose this task: A while back, I participated in a hackathon (Hack-It-All) where I used sklearn and LLMs to create specific recommendations for users, and I loved every second of the learning and development process. As soon as I saw the task description, I thought that maybe sklearn and AI could help me, and I was right.

My initial thought process: At first, when I saw the Tier 1 websites, I decided to just compare the metadata (code) between websites, and after some tweaking, I got where I wanted. The first tier was pretty easy, but the second one was where I really got AI and LLMs involved in my code because it got difficult. Metadata comparison was no longer enough due to the different languages in the code and similar aspects of the content. I decided to use tools like sklearn, TensorFlow, and Selenium to get through it all. First, the code accesses all websites with the help of ChromeDriver, a powerful tool for automation. Then, Selenium is used to take screenshots of every website. After all this, every screenshot is compared using LLMs (Sentence-BER and others) and stored separately in clusters based on similarity. The obvious choice of language for the back-end was python, due to powerful tools used by pip3, and for front end I chose Node.js.

I applied this algorithm to all tiers, with the only difference being the output clusters and website screenshot names. The code is a bit of an overkill for the first tier, but it works like a charm for every tier. That being said, my code and algorithm only focus on the visual aspect of the page because that’s what users actually see—the text isn’t important. This approach is highly scalable and can work with very large datasets and HTML files; it just needs processing time.

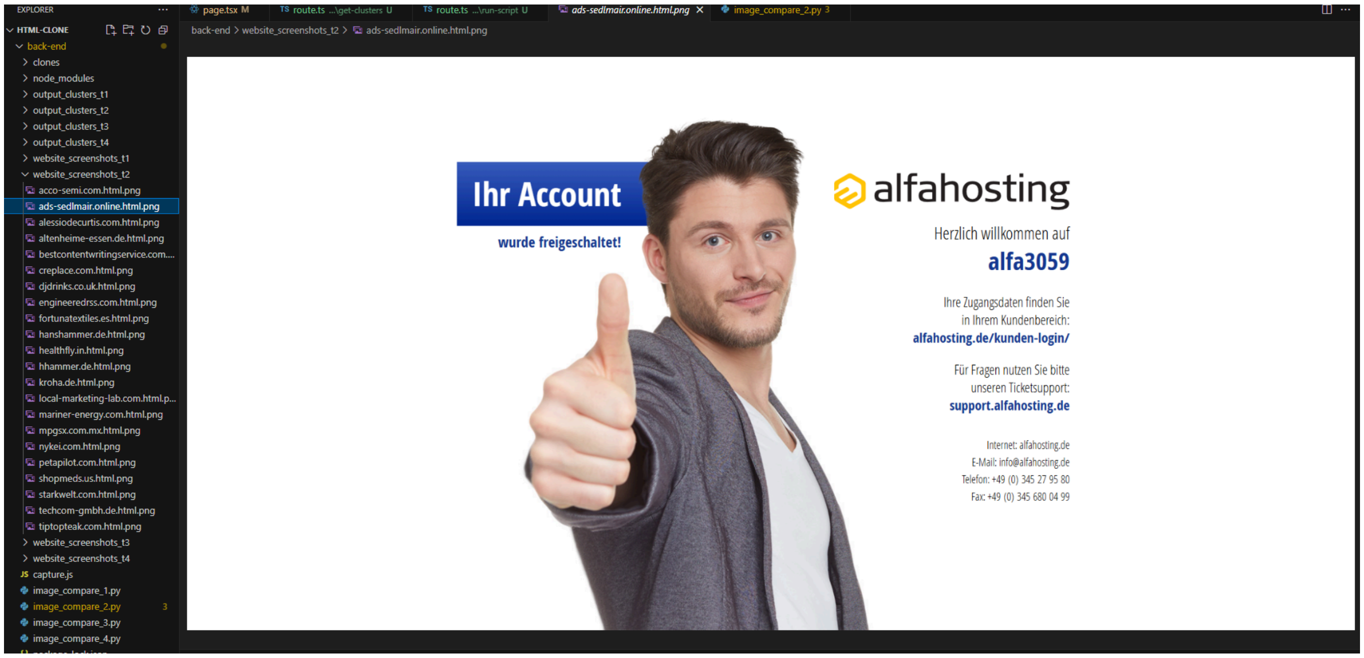
For example, it took the algorithm 60 seconds to process all 100 websites in tier 1. That’s 16 hours for 100,000 websites with the limited resources I have at home, but with better processing power, it could be reduced to just one hour for hundreds of thousands of websites. I added a front end design that generates this clusters with a press of a button, which makes the program slower because it needs to fetch information from back-end.

How I've accomplished this: As I said, at first, I wanted to compare every website with only the metadata that I have, but when tier 2 started to get more complex I needed something smarter. I will explain everything that I have done using photos of the code.



```
EXPLORER
HTML-CLONE
  back-end
    clones
    node_modules
    output_clusters_t1
    output_clusters_t2
    output_clusters_t3
    output_clusters_t4
    website_screenshots_t1
    website_screenshots_t2
    website_screenshots_t3
    website_screenshots_t4
    capture.js
    image_compare_1.py
    image_compare_2.py                    3
    image_compare_3.py
    image_compare_4.py
    package-lock.json
    package.json
    python-3.11.7-amd64.exe
    python-3.11.7.exe
  my-clustering-app
```

Back-end: This is my back-end project structure, with 4 scripts (image_compare_n.py), clones which store all the websites, website_screenshots_tn where web screenshots are stored, and output_clusters_tn where every similar website is stored in a cluster. First, the algorithm opens every website stored in a folder using chromedriver, which is an automated process whenever a function inside my code is called. When opened, it takes a screenshot that is stored in website_screenshots_tn. Then, using sklearn, TensorFlow, and selenium, each image is compared by visual similarities, just like a user sees them, and then places every website in clusters. The only drawback of this algorithm is that if the page is scrollable, the screenshot will only be of the top part of the page. Here is an example of a screenshot taken by the code:


EXPLORER
page.tsx M
route.ts ...\get-clusters U
route.ts ...\run-script U
ads-sedlmair.online.html.png
image_compare_2.py 3
back-end > website_screenshots_t2 > ads-sedlmair.online.html.png

```
HTML-CLONE
  back-end
    > clones
    > node_modules
    > output_clusters_t1
    > output_clusters_t2
    > output_clusters_t3
    > output_clusters_t4
    > website_screenshots_t1
    v website_screenshots_t2
        acco-semi.com.html.png
        ads-sedlmair.online.html.png
        alessiodecurtis.com.html.png
        altenheime-essen.de.html.png
        bestcontentwritingservice.com....
        creplace.com.html.png
        djdrinks.co.uk.html.png
        engineeredrss.com.html.png
        fortunatextiles.es.html.png
        hanshammer.de.html.png
        healthfly.in.html.png
        hhammer.de.html.png
        kroha.de.html.png
        local-marketing-lab.com.html.p...
        mariner-energy.com.html.png
        mpgsx.com.mx.html.png
        nykei.com.html.png
        petapilot.com.html.png
        shopmeds.us.html.png
        starkwelt.com.html.png
        techcom-gmbh.de.html.png
        tiptopteak.com.html.png
    > website_screenshots_t3
    > website_screenshots_t4
    capture.js
    image_compare_1.py
    image_compare_2.py          3
    image_compare_3.py
    image_compare_4.py
```

Ihr Account
wurde freigeschaltet!

alfahosting

Herzlich willkommen auf
alfa3059

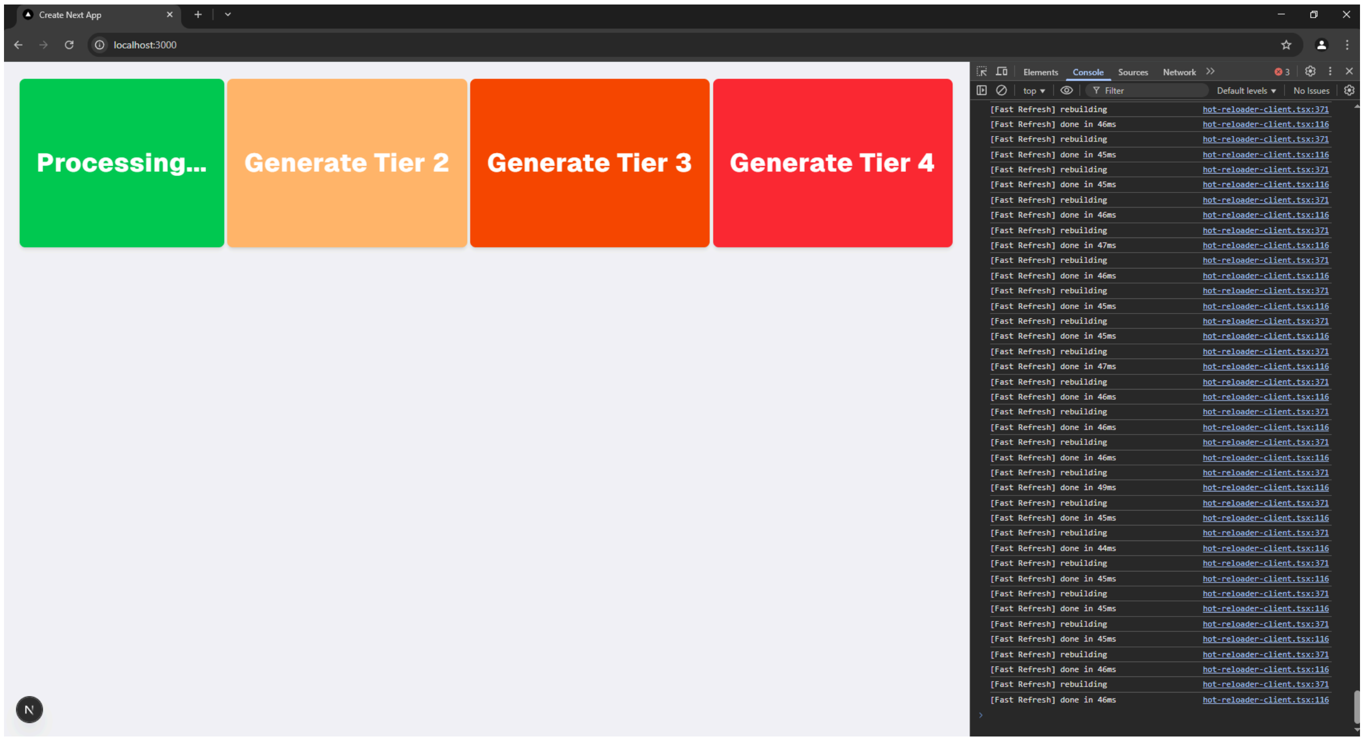
Ihre Zugangsdaten finden Sie
in Ihrem Kundenbereich:
alfahosting.de/kunden-login/

Für Fragen nutzen Sie bitte
unseren Ticketsupport:
support.alfahosting.de

Internet: alfahosting.de
E-Mail: info@alfahosting.de
Telefon: +49 (0) 345 27 95 80
Fax: +49 (0) 345 680 04 99

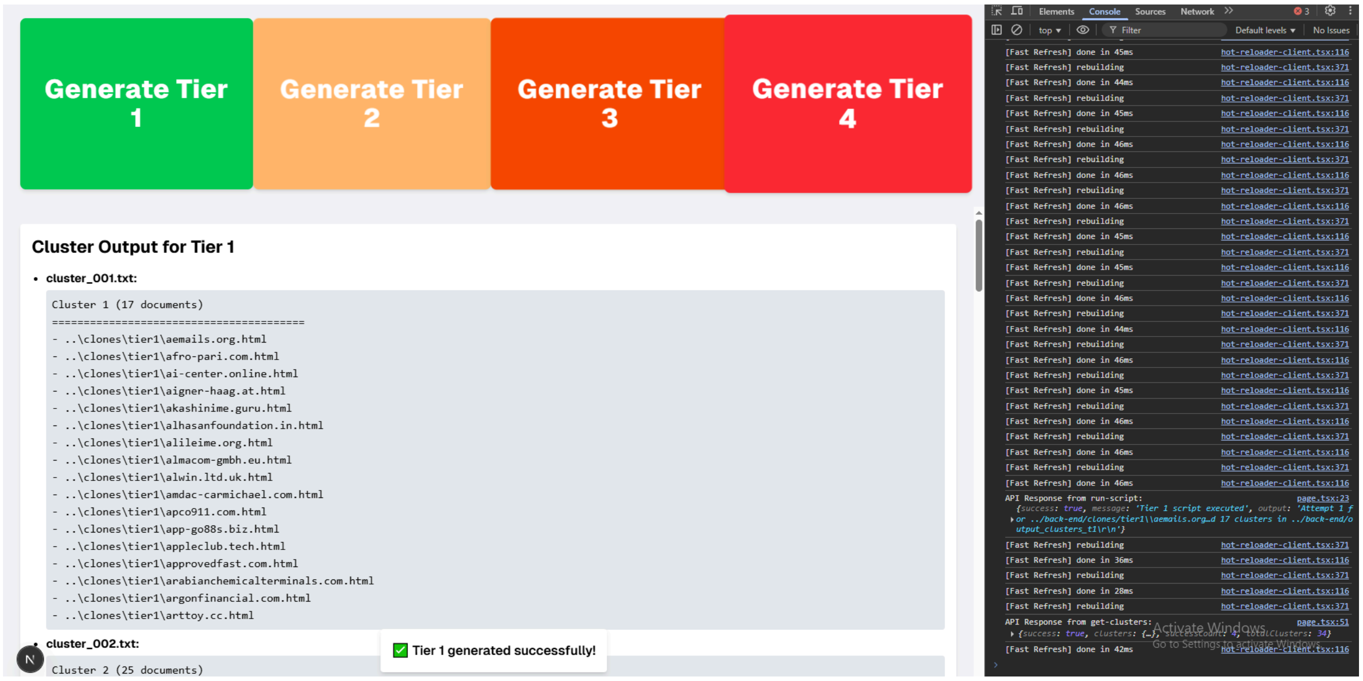
Everything is sorted in the backend, whenever a image_compare_tn is ran it generated 2 things: website_screenshots_tn and output_clusters_tn. Every image_compare_tn is exactly the same, the only change is the websites is testing.

Front-end: I really wanted to add a visual representation of the algorithm that any user can use without the need of calling functions and running dev tools in code editor, so I made a Node app:



```
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 47ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 47ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 49ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 44ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
[Fast Refresh] rebuilding           hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms         hot-reloader-client.tsx:116
```

There are 4 buttons, each generating clusters for tier 1, 2, 3 and 4. When one button is pressed, a function is called that runs the image_compare_tn designated to the button number. The code runs in background and after generating it displays the results.



Cluster Output for Tier 1

- **cluster_001.txt:**

```
Cluster 1 (17 documents)
========================================
- ..\clones\tier1\aemails.org.html
- ..\clones\tier1\afro-pari.com.html
- ..\clones\tier1\ai-center.online.html
- ..\clones\tier1\aigner-haag.at.html
- ..\clones\tier1\akashinime.guru.html
- ..\clones\tier1\alhasanfoundation.in.html
- ..\clones\tier1\alileime.org.html
- ..\clones\tier1\almacom-gmbh.eu.html
- ..\clones\tier1\alwin.ltd.uk.html
- ..\clones\tier1\amdac-carmichael.com.html
- ..\clones\tier1\apco911.com.html
- ..\clones\tier1\app-go88s.biz.html
- ..\clones\tier1\appleclub.tech.html
- ..\clones\tier1\approvedfast.com.html
- ..\clones\tier1\arabianchemicalterminals.com.html
- ..\clones\tier1\argonfinancial.com.html
- ..\clones\tier1\arttoy.cc.html
```

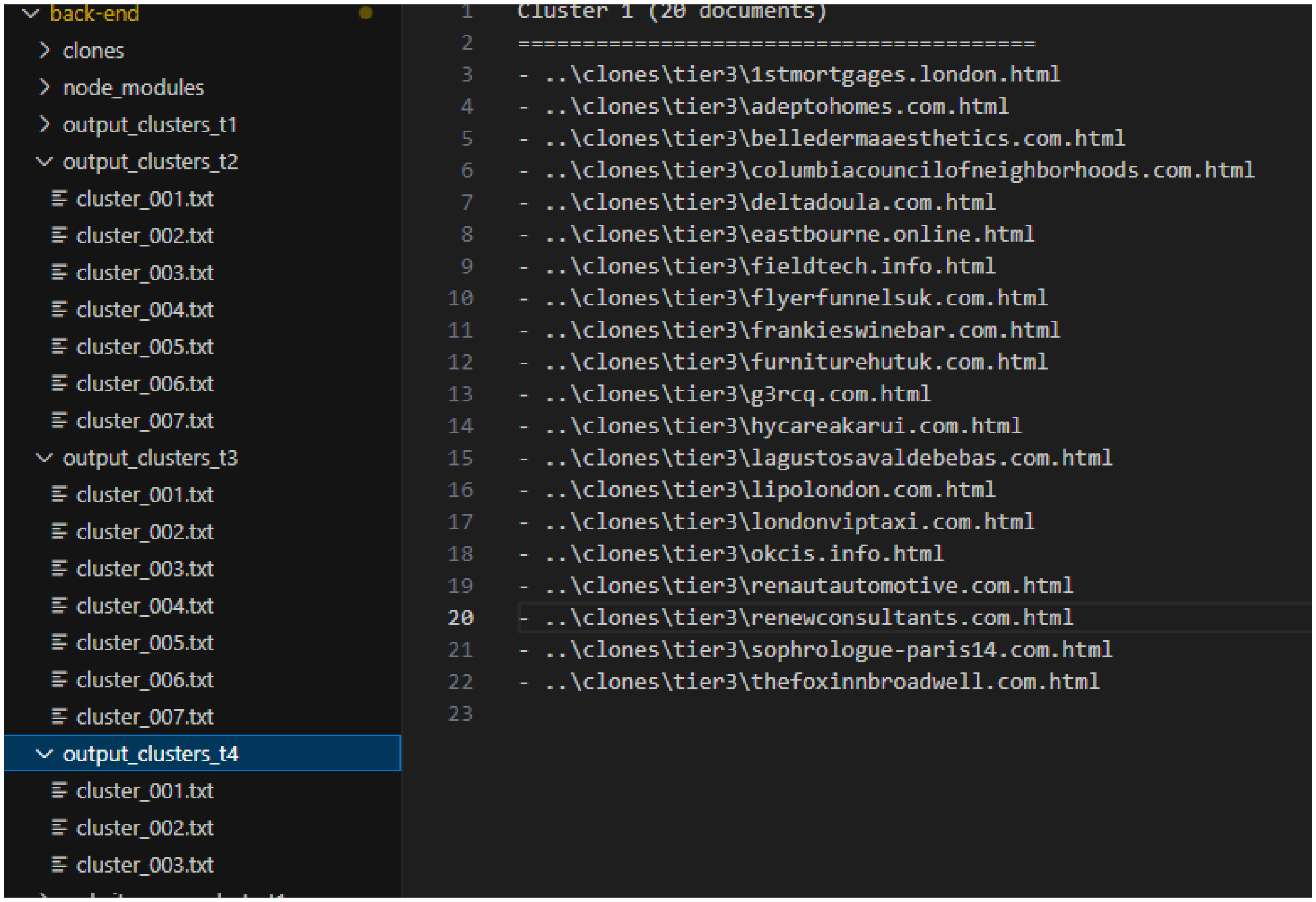
- **cluster_002.txt:**

```
Cluster 2 (25 documents)
```

✅ Tier 1 generated successfully!

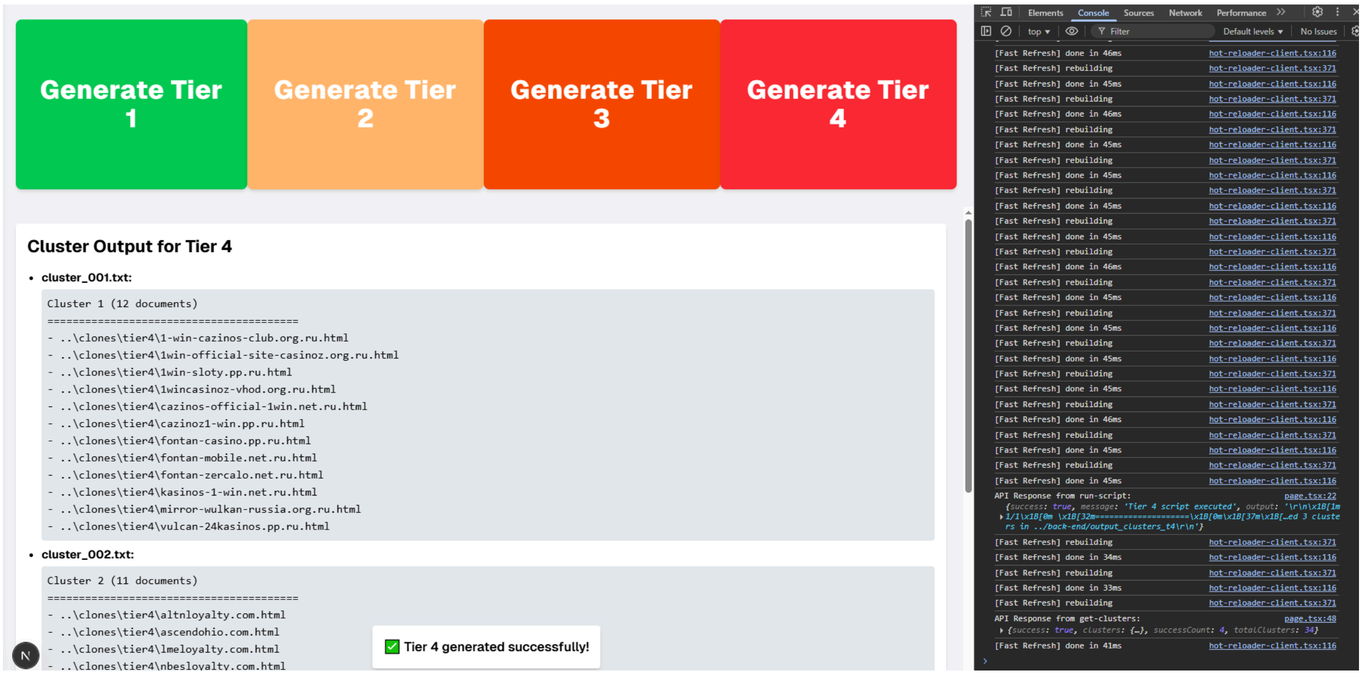
```
[Fast Refresh] done in 45ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 44ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 44ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                    hot-reloader-client.tsx:116
API Response from run-script:                                page.tsx:23
  {success: true, message: 'Tier 1 script executed', output: 'Attempt 1 f
 ▸or ../back-end/clones/tier1\\aemails.org…d 17 clusters in ../back-end/o
  utput_clusters_t1\r\n'}
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 36ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
[Fast Refresh] done in 28ms                    hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                      hot-reloader-client.tsx:371
API Response from get-clusters:                              page.tsx:51
 ▸{success: true, clusters: {…}, successCount: 4, totalClusters: 34}
[Fast Refresh] done in 42ms                    hot-reloader-client.tsx:116
```

For example this are the results for the first tier, and this is what is saved in the project:



```
Cluster 1 (20 documents)
========================================
- ..\clones\tier3\1stmortgages.london.html
- ..\clones\tier3\adeptohomes.com.html
- ..\clones\tier3\belledermaaesthetics.com.html
- ..\clones\tier3\columbiacouncilofneighborhoods.com.html
- ..\clones\tier3\deltadoula.com.html
- ..\clones\tier3\eastbourne.online.html
- ..\clones\tier3\fieldtech.info.html
- ..\clones\tier3\flyerfunnelsuk.com.html
- ..\clones\tier3\frankieswinebar.com.html
- ..\clones\tier3\furniturehutuk.com.html
- ..\clones\tier3\g3rcq.com.html
- ..\clones\tier3\hycareakarui.com.html
- ..\clones\tier3\lagustosavaldebebas.com.html
- ..\clones\tier3\lipolondon.com.html
- ..\clones\tier3\londonviptaxi.com.html
- ..\clones\tier3\okcis.info.html
- ..\clones\tier3\renautautomotive.com.html
- ..\clones\tier3\renewconsultants.com.html
- ..\clones\tier3\sophrologue-paris14.com.html
- ..\clones\tier3\thefoxinnbroadwell.com.html
```

And this, is for the 4th tier:



Cluster Output for Tier 4

- **cluster_001.txt:**

```
Cluster 1 (12 documents)
========================================
- ..\clones\tier4\1-win-cazinos-club.org.ru.html
- ..\clones\tier4\1win-official-site-casinoz.org.ru.html
- ..\clones\tier4\1win-sloty.pp.ru.html
- ..\clones\tier4\1wincasinoz-vhod.org.ru.html
- ..\clones\tier4\cazinos-official-1win.net.ru.html
- ..\clones\tier4\cazinoz1-win.pp.ru.html
- ..\clones\tier4\fontan-casino.pp.ru.html
- ..\clones\tier4\fontan-mobile.net.ru.html
- ..\clones\tier4\fontan-zercalo.net.ru.html
- ..\clones\tier4\kasinos-1-win.net.ru.html
- ..\clones\tier4\mirror-wulkan-russia.org.ru.html
- ..\clones\tier4\vulcan-24kasinos.pp.ru.html
```

- **cluster_002.txt:**

```
Cluster 2 (11 documents)
========================================
- ..\clones\tier4\altnloyalty.com.html
- ..\clones\tier4\ascendohio.com.html
- ..\clones\tier4\lmeloyalty.com.html
- ..\clones\tier4\nbesloyalty.com.html
```

✅ Tier 4 generated successfully!

```
[Fast Refresh] done in 46ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 46ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 45ms                     hot-reloader-client.tsx:116
API Response from run-script:                                 page.tsx:22
  {success: true, message: 'Tier 4 script executed', output: '\r\n\x1B[1m
 ▸1/1\x1B[0m \x1B[32m====================\x1B[0m\x1B[37m\x1B[…ed 3 cluste
  rs in ../back-end/output_clusters_t4\r\n'}
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 34ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
[Fast Refresh] done in 33ms                     hot-reloader-client.tsx:116
[Fast Refresh] rebuilding                       hot-reloader-client.tsx:371
API Response from get-clusters:                               page.tsx:48
 ▸{success: true, clusters: {…}, successCount: 4, totalClusters: 34}
[Fast Refresh] done in 41ms                     hot-reloader-client.tsx:116
›
```

And this is it: In conclusion, this app access every html provided, takes a screenshot, compare it to others and put them in clusters. I really loved developing this app, I really hope it raises to the expectations and I want to take for the opportunity and for making such interesting task!

References:

1. LLM's like chatgpt and deepseek
2. Stackoverflow
3. ChromeForum
4. Selenium Documentation
5. Clustering Algorithms (DBSCAN)
6. My mind:)