

# **Life of a personal object README**

For my personal object scrolltelling website, i will display some of my drawings and a sequence of a painting that represent its transformation. I will use gsap scroll trigger method to set a specific scroll effect, that pins the page to the top. I will also add a theme picker that changes between light and dark mode, to increase the accessibility of my webpage.

## **HTML sections**

In this project I will have slide-like elements that I will position as section tag in html. Each new section will equal to viewport, so I will have 5 or so slides that go one after another.

I will have my button at the very top, but I'll get to that later

On my intro page I will paste a heading one, which will be my main title and some text  
For the design purposes I'll be using lorem ipsum paragraphs along the way just to see how the sample text would look. At the final stage of the project I will swap my text with the one I have written for this weekly task.

## **CSS styling**

I want to use darkmode and light mode in this project to increase accessibility, so I define my default and darkmode class in css.

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar has icons for Explorer, Search, Find, and Outline. The Explorer panel shows a project folder named 'WEEKLY TASK - GSAP ANIMATION' containing files 'app.js', 'index.html', and '# style.css'. The '# style.css' file is currently open in the main editor area. The code defines a root style and a darkmode style, both using CSS variables for colors. The code is as follows:

```
# style.css > .darkmode
1 /* I will create the darkmode option for this website to increase the readability */
2
3 :root {
4   --base-color: #rgb(239, 193, 134);
5   --base-variant: #rgb(238, 177, 96);
6   --primary-text: #rgb(145, 36, 22);
7   --secondary-text: #rgb(22, 20, 20);
8 }
9
10 .darkmode {
11   --base-color: #rgb(22, 20, 20);
12   --base-variant: #rgb(55, 49, 49);
13   --primary-text: #rgb(255, 255, 255);
14   --secondary-text: #rgb(199, 207, 239);
15 }
16
17 *, *::after, *::before {
18   box-sizing: border-box;
19   font-family: "Inter", sans-serif;
20   font-size: 1.75rem;
21   font-weight: bold;
22 }
```

The bottom status bar shows the file path as 'Ln 10, Col 1 (167 selected)', encoding as 'UTF-8', and other details like 'Spaces: 4' and 'LF'.

It means that whenever I will introduce a new color, I will have to use a variable, and define it in my root and darkmode sections.

I have made some just to see how the effect would look like.

I made a search in how to set darkmode in vanilla CSS and Javascript to get some ideas on how to do it. I found this ('Create a Dark Mode Switch with HTML, CSS, JavaScript', 2024) reference specifically helpful and I have used it as a base for my code for the light mode / dark mode switch button, with some changes I have made.

As I have defined my themes, the next step is to make UI for the user to control switch of the themes.

I will use button html tag and style it in CSS

This screenshot shows the Google Fonts website's icon customization tool. On the left, there's a sidebar with categories like Fonts, Icons, and Knowledge. The main area has sections for 'Customize' (with 'Reset all'), 'Your Privacy and Google Fonts', and 'Google Fonts'. A search bar says 'dark mode'. To the right, there's a 'Dark Mode' section with a preview icon, size (24), color (#E8EAD), download buttons for SVG and PNG, and tabs for Web, Android, and Apple. Below this is an 'Instructions' section and a 'Variable icon font' section with code examples.

This screenshot shows a web browser window with a page titled "Andrii's Personal Object - Paintings". The page content includes a paragraph of placeholder text (Lorem ipsum) and a small dark mode icon in the top right corner. The browser's address bar shows the file path: "/Users/andriartemenko/Desktop/Weekly%20Tasks%20WebDev%20Studio%20Submission/Weekly%20Task%20-%20GSAP%20".

```
# style.css > .BTN
70    width: 100px;
71    height: 50px;
72    cursor: pointer;
73    border: none;
74    border-radius: 25px;
75    display:flex;
76    justify-content: center;
77    align-items: center;
78 }
79
80 .BTN svg:last-child{
81    display:none;
82 }
83
84 .darkmode .BTN svg:first-child{
85    display: none;
86 }
87
88 .darkmode .BTN svg:last-child{
89    display:flex;
90 }
91
```

Here, I learned how I can switch the content of the svg container based on whether darkmode is applied or no. Because I have 2 'children' svgs of a class BTN, i can refer to the at first and last child. I will later add interactivity in javascript, but so far it defines some styling changes.

In my JS file I can access my darkmode class through the localStorage getItem method.

I can get manipulate my button through DOM by method getElementById.

I add an event listener on click for my button, that will display dark or light mode button, and switch the themes accordingly.

First, I make an if statement condition to check if darkmode is active. If it is active, the function enableDarkmode will run, and if not it will reverse it, which will switch it back to light mode.

The screenshot shows the Visual Studio Code (VS Code) interface with a dark theme. The left sidebar contains icons for Explorer, Search, Open, and Outline/Timeline. The Explorer view shows a project folder named 'WEEKLY TASK - GSAP ANIMATION' containing 'assets' (with 'dark\_mode.svg'), 'index.html', '# style.css', and 'JS app.js'. The 'app.js' file is open in the main editor, displaying the following code:

```
JS app.js > themeSwitch.addEventListener("click") callback
1 let darkmode = localStorage.getItem('darkmode')
2 const themeSwitch = document.getElementById('theme-switch')
3
4 themeSwitch.addEventListener("click", () => {
5   if(darkmode !== "active"){
6     enableDarkmode()
7   }
8   else{
9     disableDarkmode()
10  }
11})
```

The editor tabs at the top show 'Welcome', 'index.html', 'dark\_mode.svg', '# style.css', and 'JS app.js'. Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE' (which is selected), 'TERMINAL', and 'PORTS'. A search bar says 'Filter (e.g. text, !exclu...)' and a status bar at the bottom shows 'Ln 9, Col 26 Spaces: 4 UTF-8 LF {JavaScript} Go Live Pr'.

The enable Darkmode function will simply add darkmode class to the list of all classes, , and update the local storage.

The screenshot shows the Visual Studio Code (VS Code) interface with a dark-themed UI. The top bar displays the title "Weekly Task - GSAP Animation". The left sidebar contains icons for Explorer, Search, Open, Find, Outline, Timeline, and a preview area. The Explorer view shows a project structure with a folder named "WEEKLY TASK - GSAP ANIMATION" containing "assets" (with "dark\_mode.svg"), "app.js", "index.html", and "style.css". The "app.js" file is currently open in the main editor tab, showing the following code:

```
JS app.js > ...
1 let darkmode = localStorage.getItem('darkmode')
2 const themeSwitch = document.getElementById('theme-switch')
3
4 const enableDarkmode = () => {
5   document.body.classList.add('darkmode')
6   localStorage.setItem('darkmode', 'active')
7 }
8
9 themeSwitch.addEventListener("click", () => {
10   if(darkmode !== "active"){
11     enableDarkmode()
12   }
13   else{
14     disableDarkmode()
15   }
16 })
```

The editor tabs at the top also include "Welcome", "index.html", "dark\_mode.svg", "# style.css", and "app.js". Below the editor, there are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE" (which is selected), "TERMINAL", and "PORTS". A search bar says "Filter (e.g. text, exclude...)". At the bottom, status information includes "Ln 4, Col 1 (123 selected)", "Spaces: 4", "UTF-8", "LF", "JavaScript", "Go Live", and "Pre".

```
let darkmode = localStorage.getItem('darkmode')
const themeSwitch = document.getElementById('theme-switch')

const enableDarkmode = () => {
    document.body.classList.add('darkmode')
    localStorage.setItem('darkmode', 'active')
}

const disableDarkmode = () => {
    document.body.classList.remove('darkmode')
    localStorage.setItem('darkmode', null)
}

themeSwitch.addEventListener("click", () => {
    if(darkmode !== "active"){
        enableDarkmode()
    } else{
        disableDarkmode()
    }
})
```

The screenshot shows a dark-themed code editor interface. The left sidebar has icons for file, search, and refresh. The Explorer panel shows a project folder 'WEEKLY TASK - GSAP ANIMATION' containing 'assets' (with 'dark\_mode.svg'), 'index.html', and 'style.css'. The main editor area shows a JavaScript file 'app.js' with the code provided above. Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE' (which is selected), 'TERMINAL', and 'PORTS'. A search bar at the bottom says 'Filter (e.g. text, !excl...)'.

disableDarkmode does completely the opposite, it removes the darkmode class and also updates the localStorage.

One of the limitations of setItem method is that only strings can be stored in it, so instead of boolean type we have to store 'active'

## Further CSS Styling

I will use previous knowledge gained in building CSS to construct a 4 column layout for each section of my website.

Firstly, I loaded some content to see how it looks and used CSS functions like object-fit, transform(), align-center etc. to position the elements in the right places

tailwindcss.com/docs/object-fit

Today, 18:00

your record

ndcss v4.0

on-break

Ut

C

ol

ol

ol

ol

ol

ol

ol

ol

EX

D

RE

OUTLINE

TIMELINE

EXPLORER

WEEKLY TASK - GSAP ANIMATION

assets

dark\_mode.svg

Slide2\_element1.jpg

app.js

index.html

# style.css

elcome

index.html

dark\_mode.svg

# style.css

Slide2\_element1.jpg

```
# style.css > .Pic {  
 40 }  
41  
42 .Slide2 {  
 43   background-color: var(--base-variant);  
 44   position: relative;  
 45   display: flex;  
 46   justify-content: center;  
 47   align-items: center;  
 48 }  
49  
50 .Pic {  
 51   position: absolute;  
 52   object-fit: contain;  
 53 }  
54  
55 .Pic img{  
 56   object-fit: contain;  
 57 }  
58  
59 .Intro_H1 {  
 60   color: var(--primary-text);  
 61   font-size: 2.25rem;
```

PROBLEMS

OUTPUT

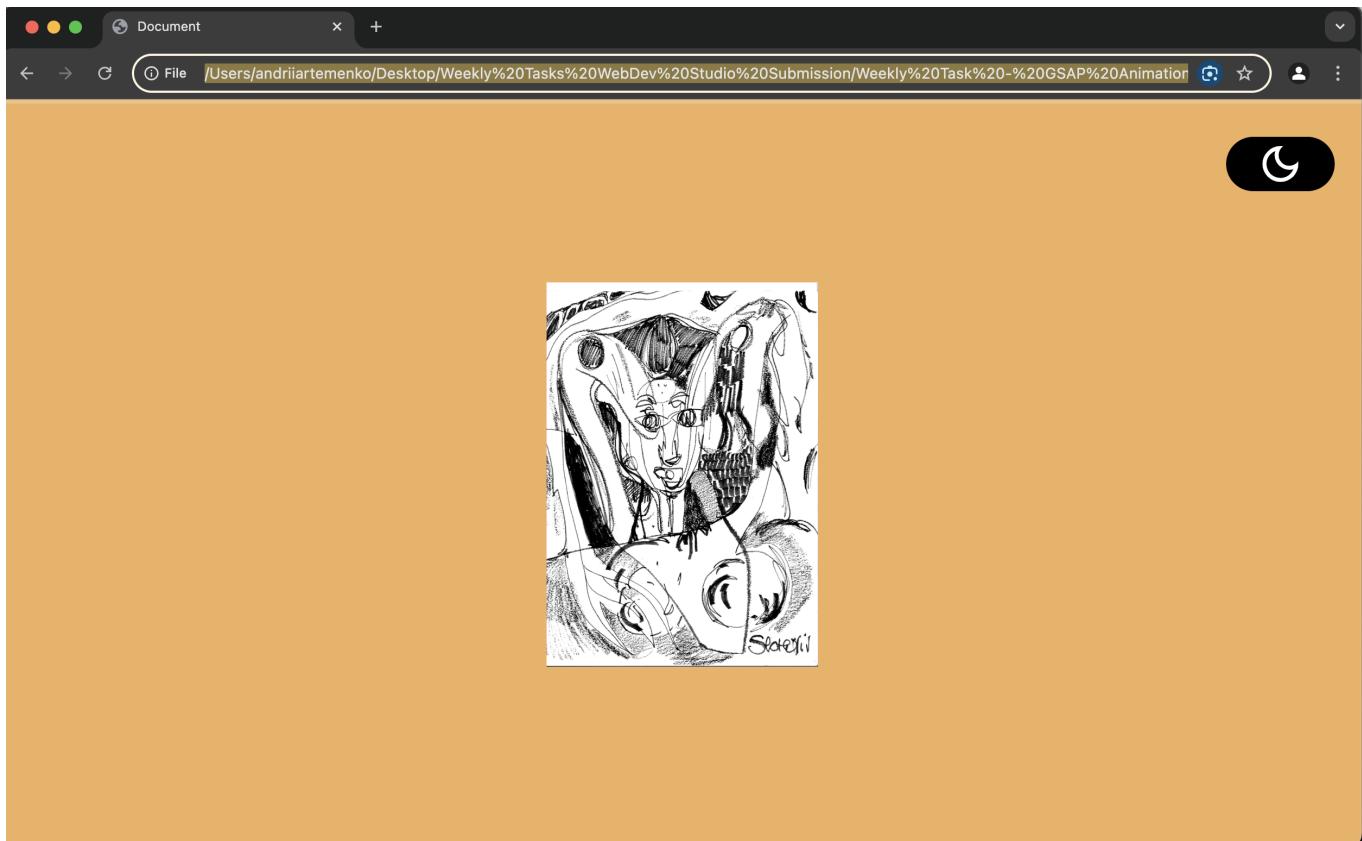
DEBUG CONSOLE

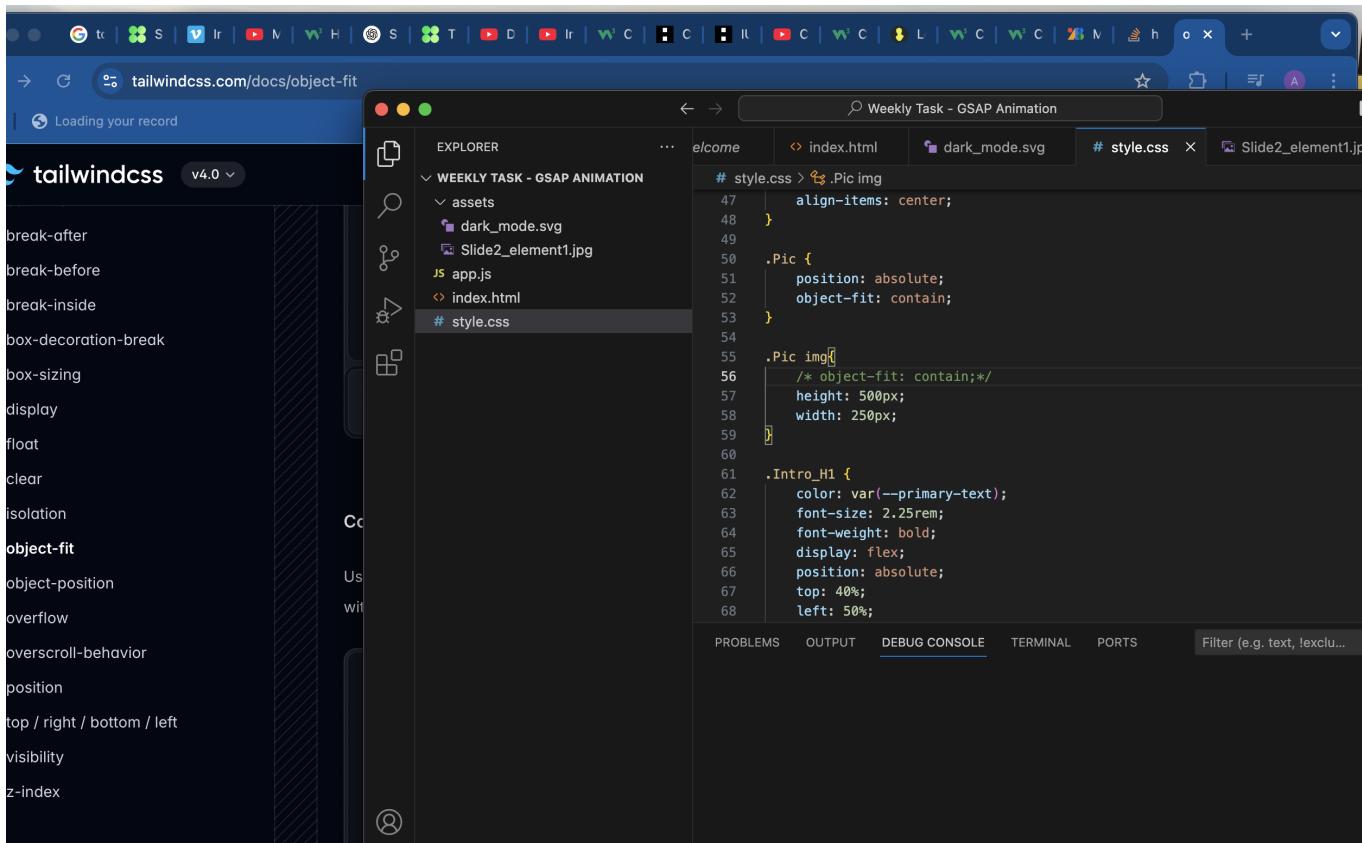
TERMINAL

PORTS

Filter (e.g. text, exclude...)

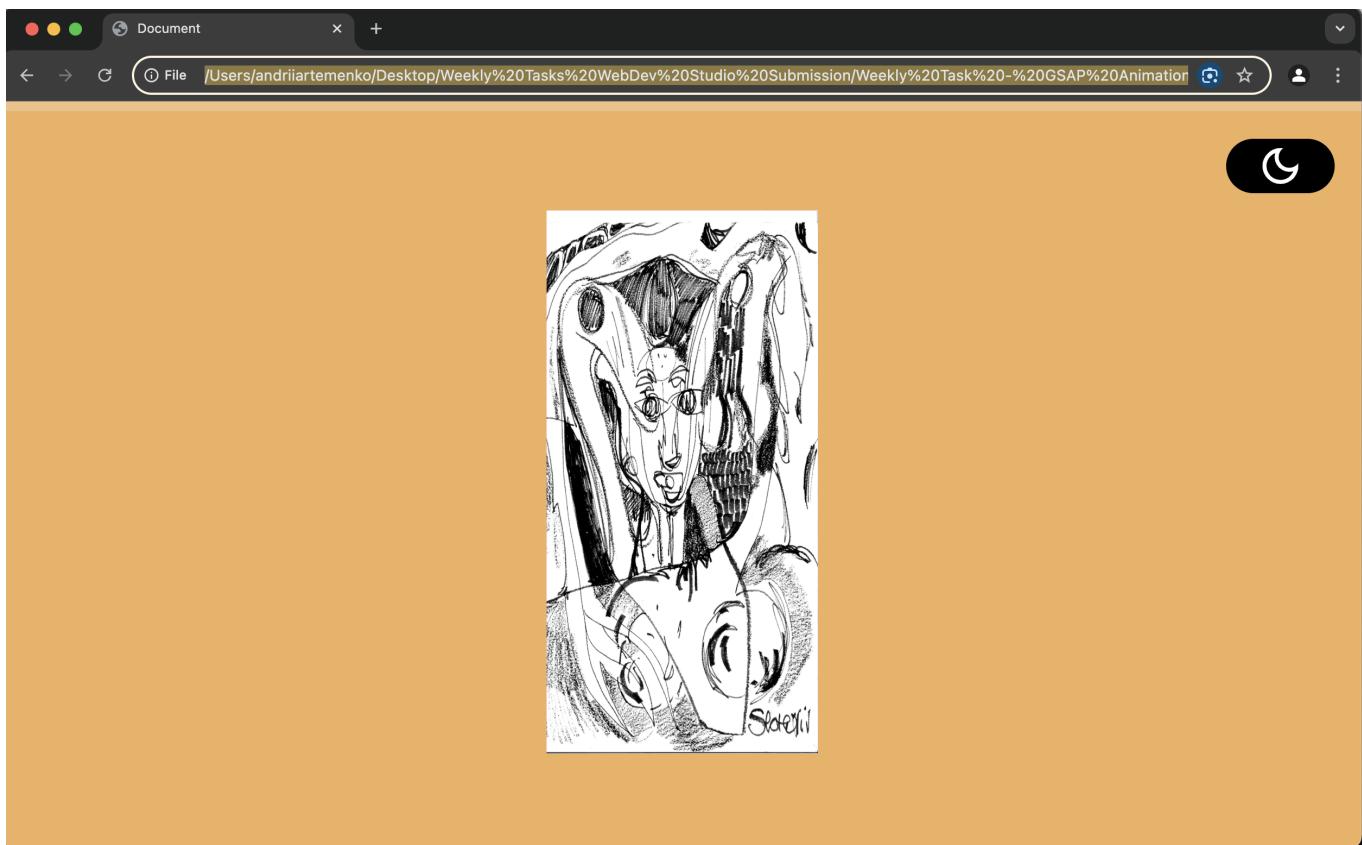
object-fit: contain; is important, because it helps to display the image nicely in a pic container



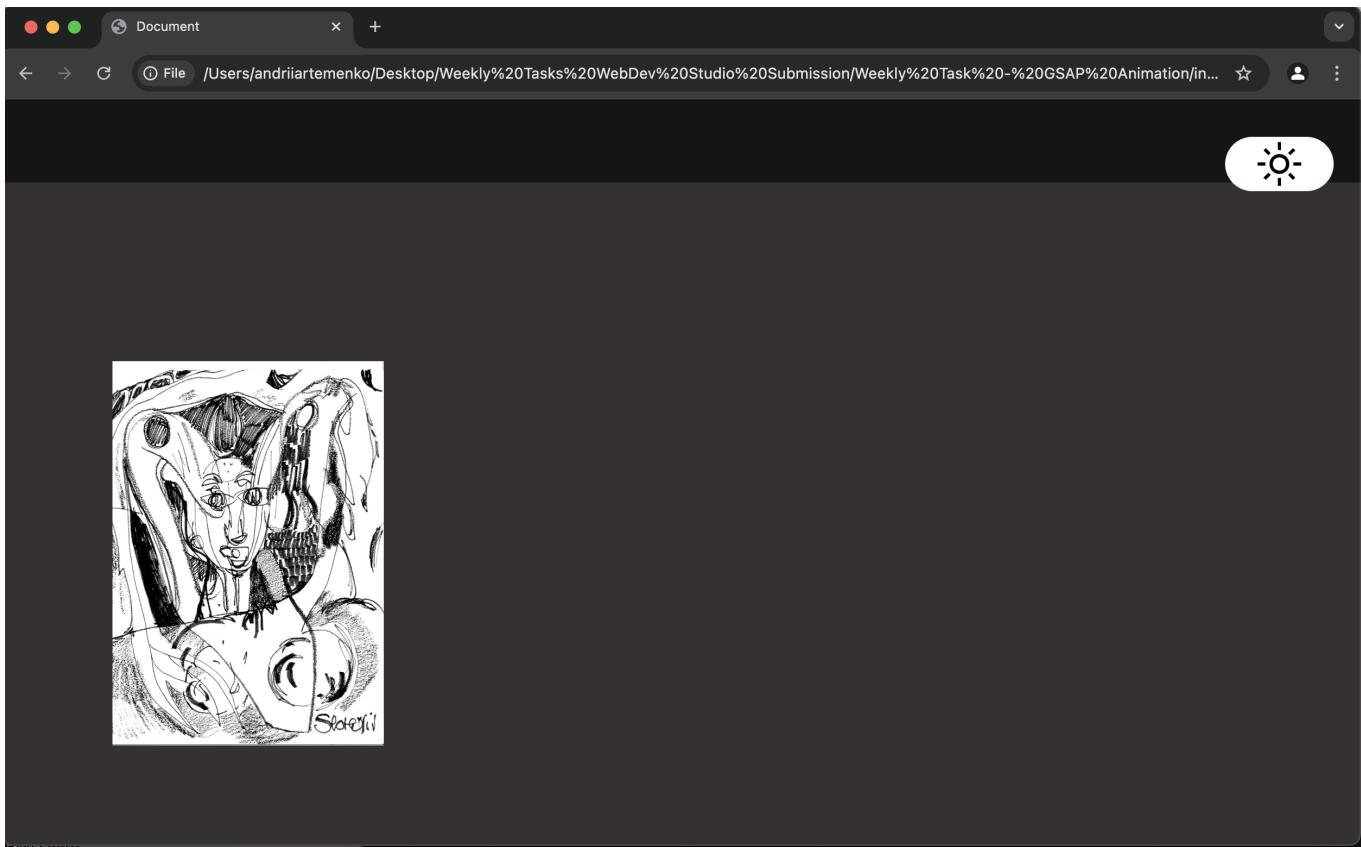


The screenshot shows a browser window with the URL [tailwindcss.com/docs/object-fit](https://tailwindcss.com/docs/object-fit). The page displays a list of Tailwind CSS utility classes on the left and a code editor on the right. The code editor contains a file named `# style.css` with the following content:

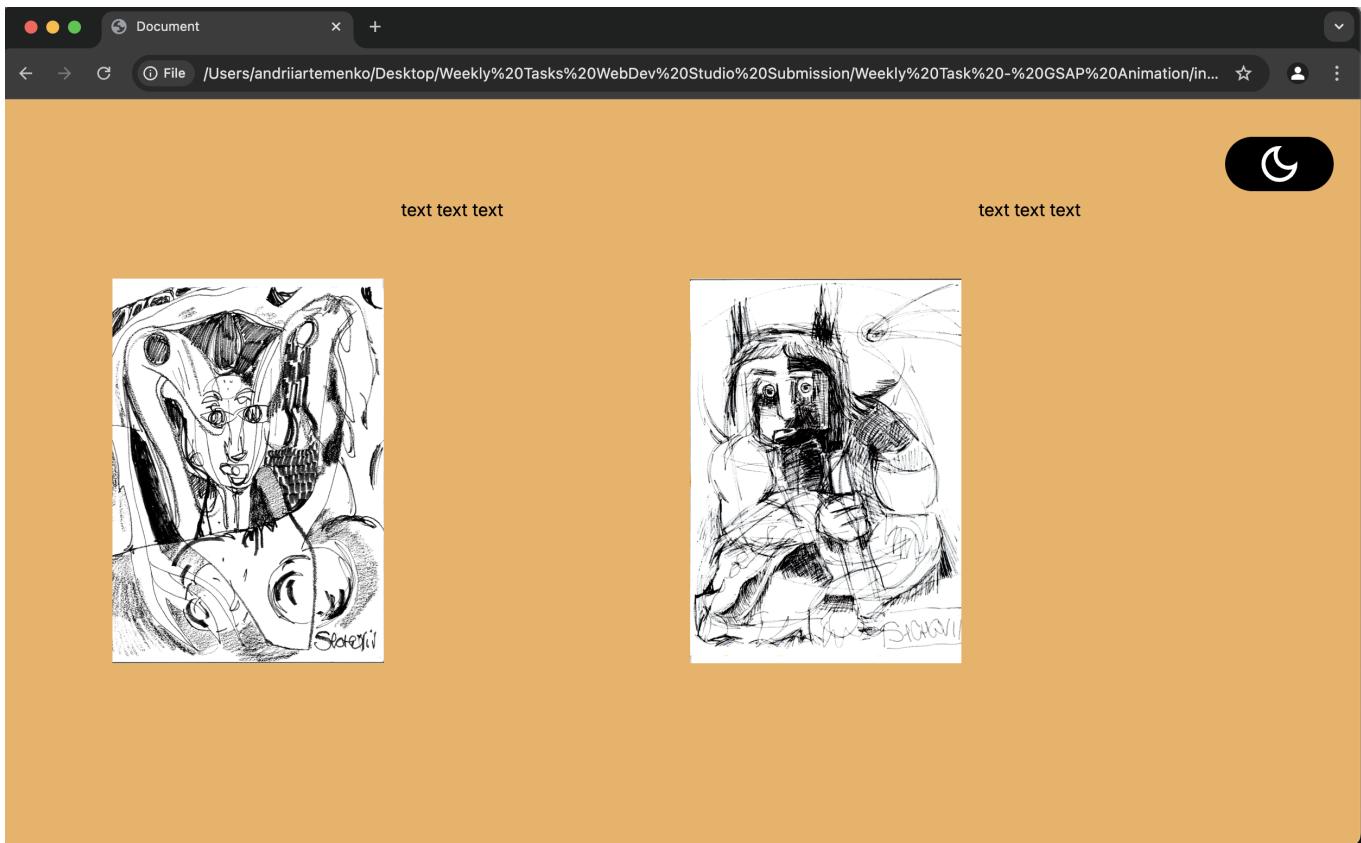
```
elcome index.html dark_mode.svg # style.css
# style.css > .Pic img
47 align-items: center;
48 }
49
50 .Pic {
51 position: absolute;
52 object-fit: contain;
53 }
54
55 .Pic img{
56 /* object-fit: contain;*/
57 height: 500px;
58 width: 250px;
59 }
60
61 .Intro_H1 {
62 color: var(--primary-text);
63 font-size: 2.25rem;
64 font-weight: bold;
65 display: flex;
66 position: absolute;
67 top: 40%;
68 left: 50%;
```

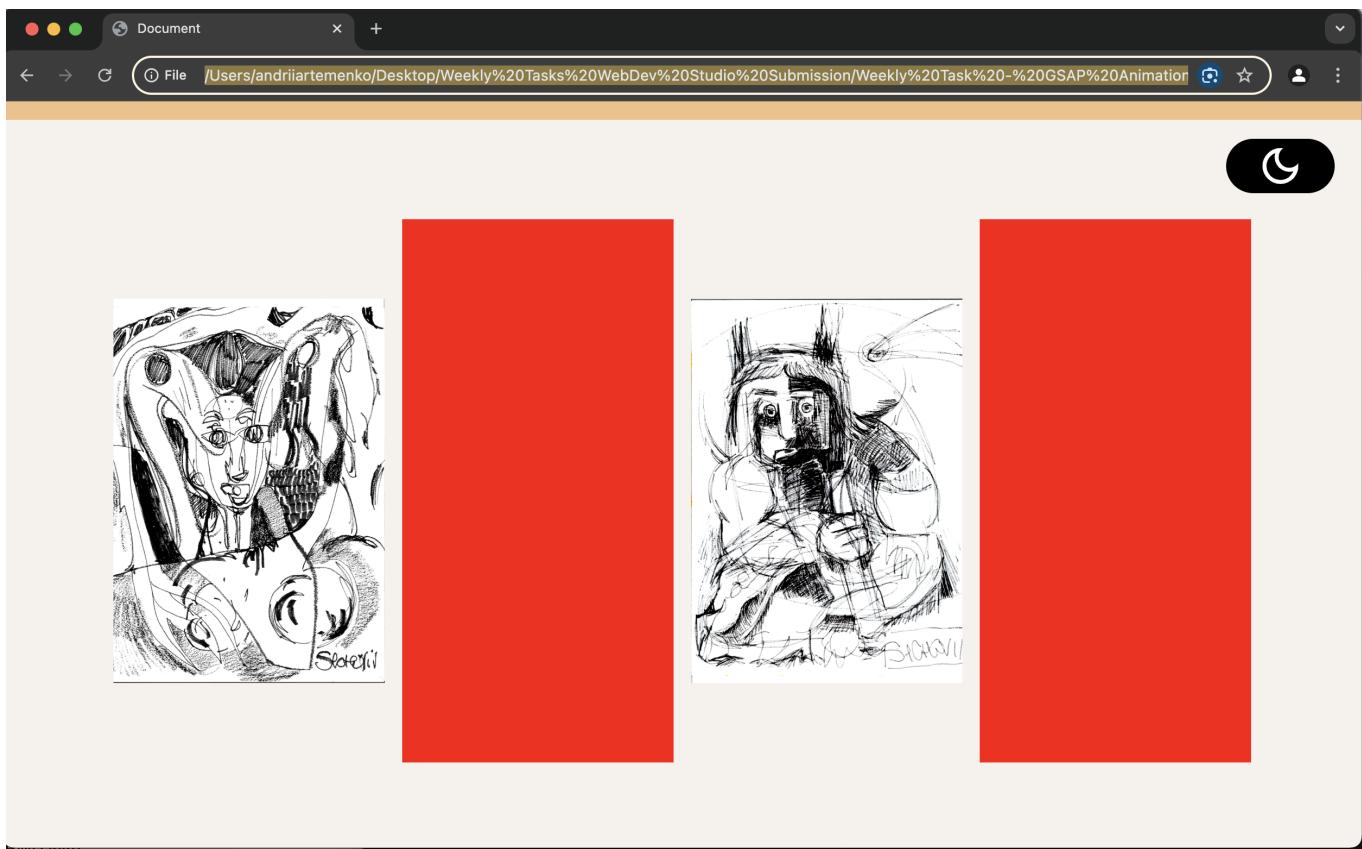


\*Object-fit: contain commented out and image gets stretched instead of nicely fitting.

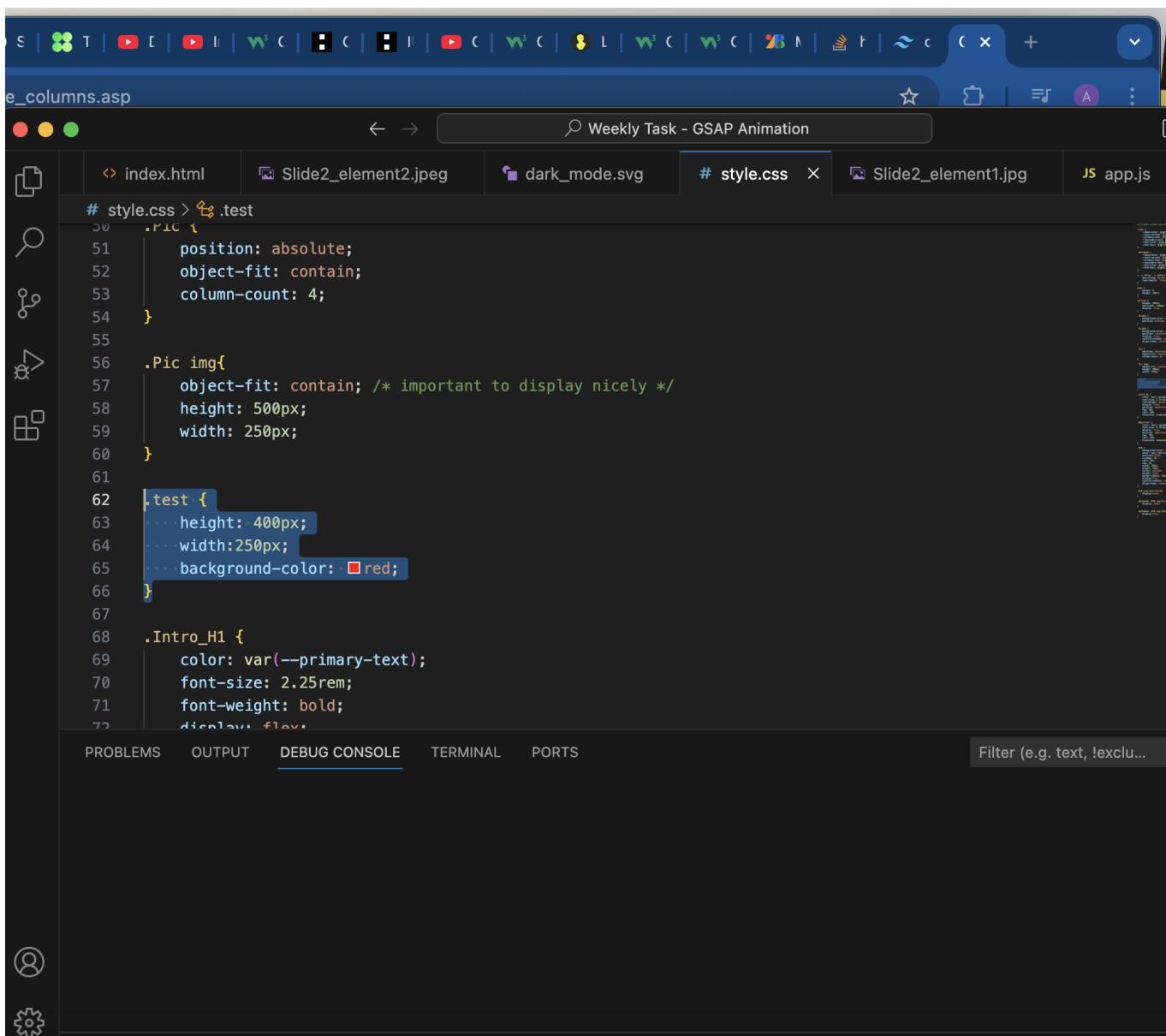


Using the 4 column layout here





I have build div containers for my headings and text, and then i will build containers within this container to separate my header sub-title and paragraph.

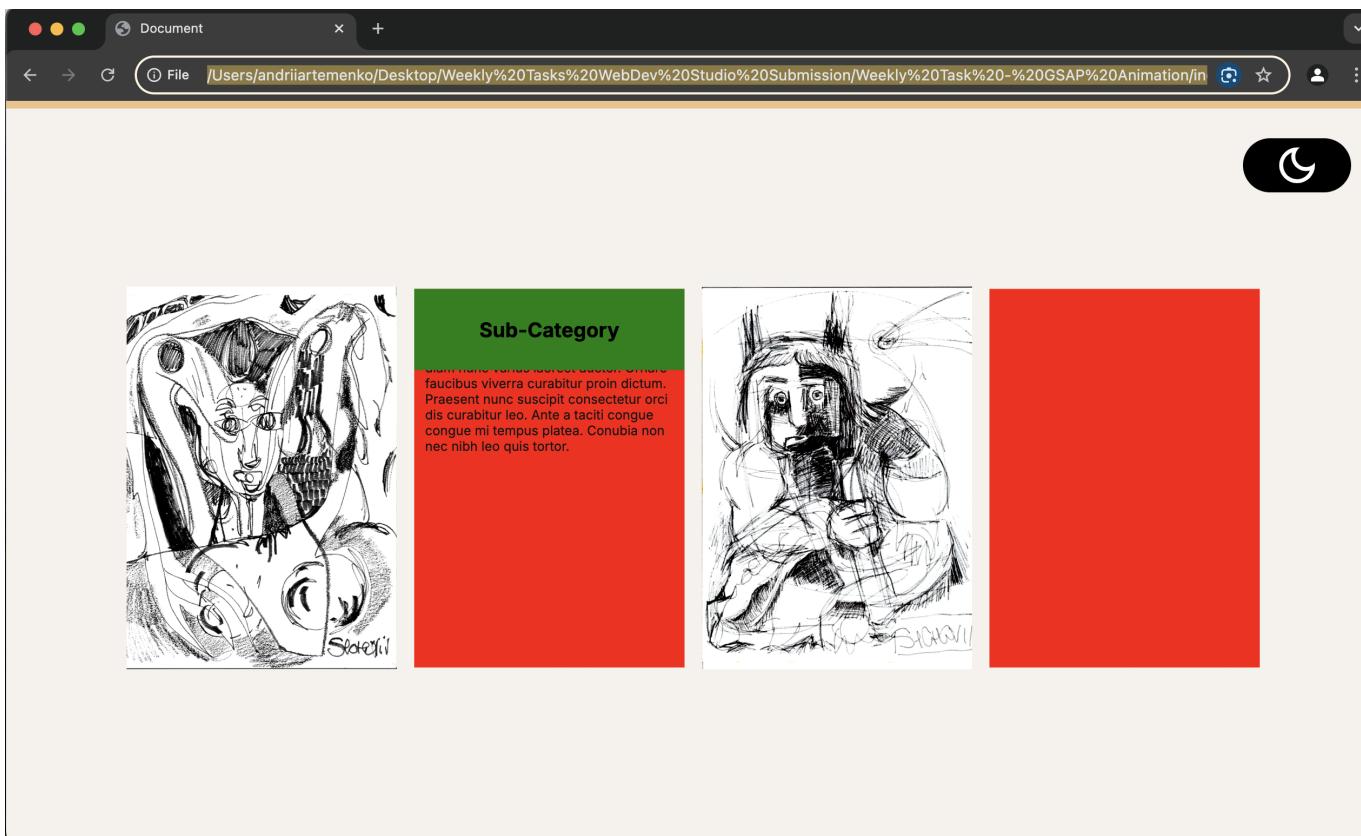
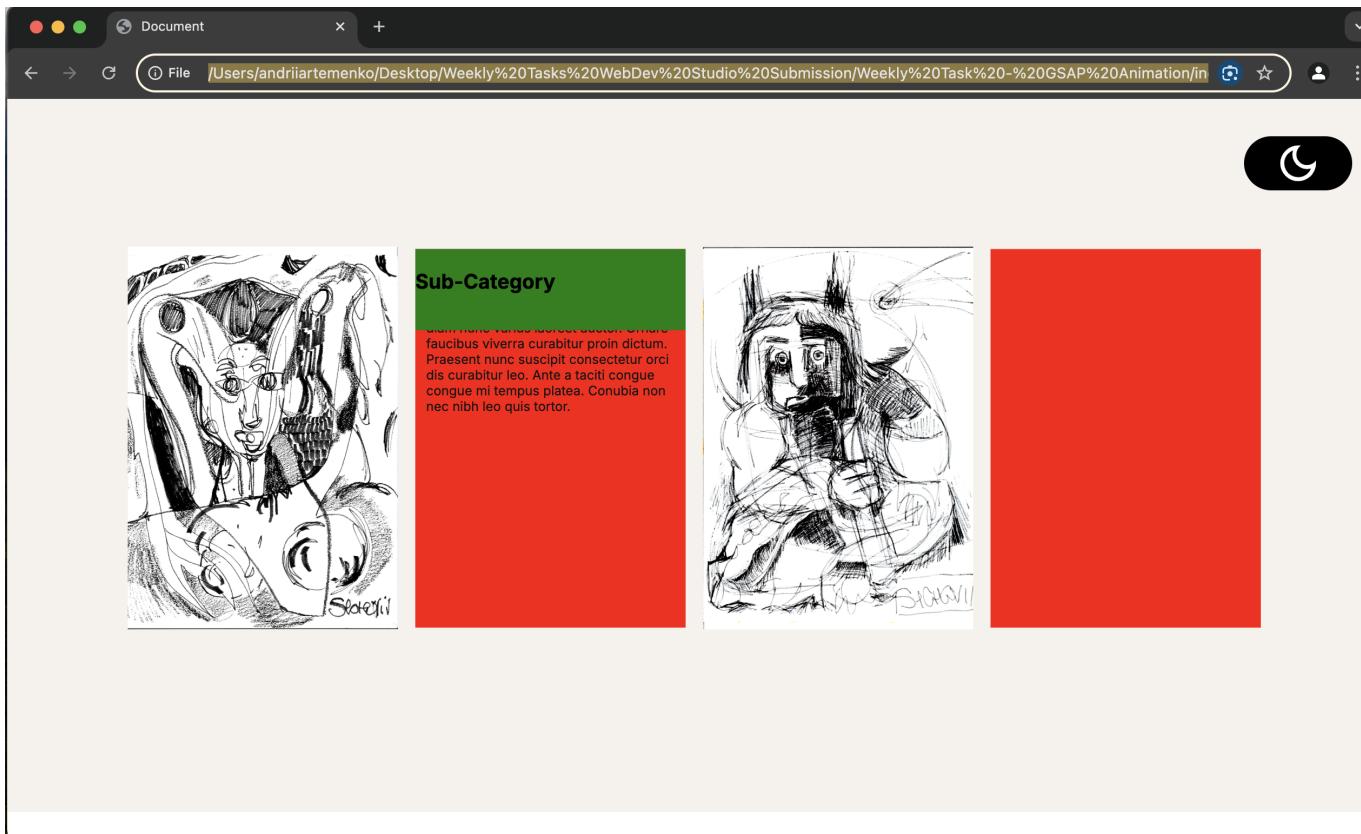


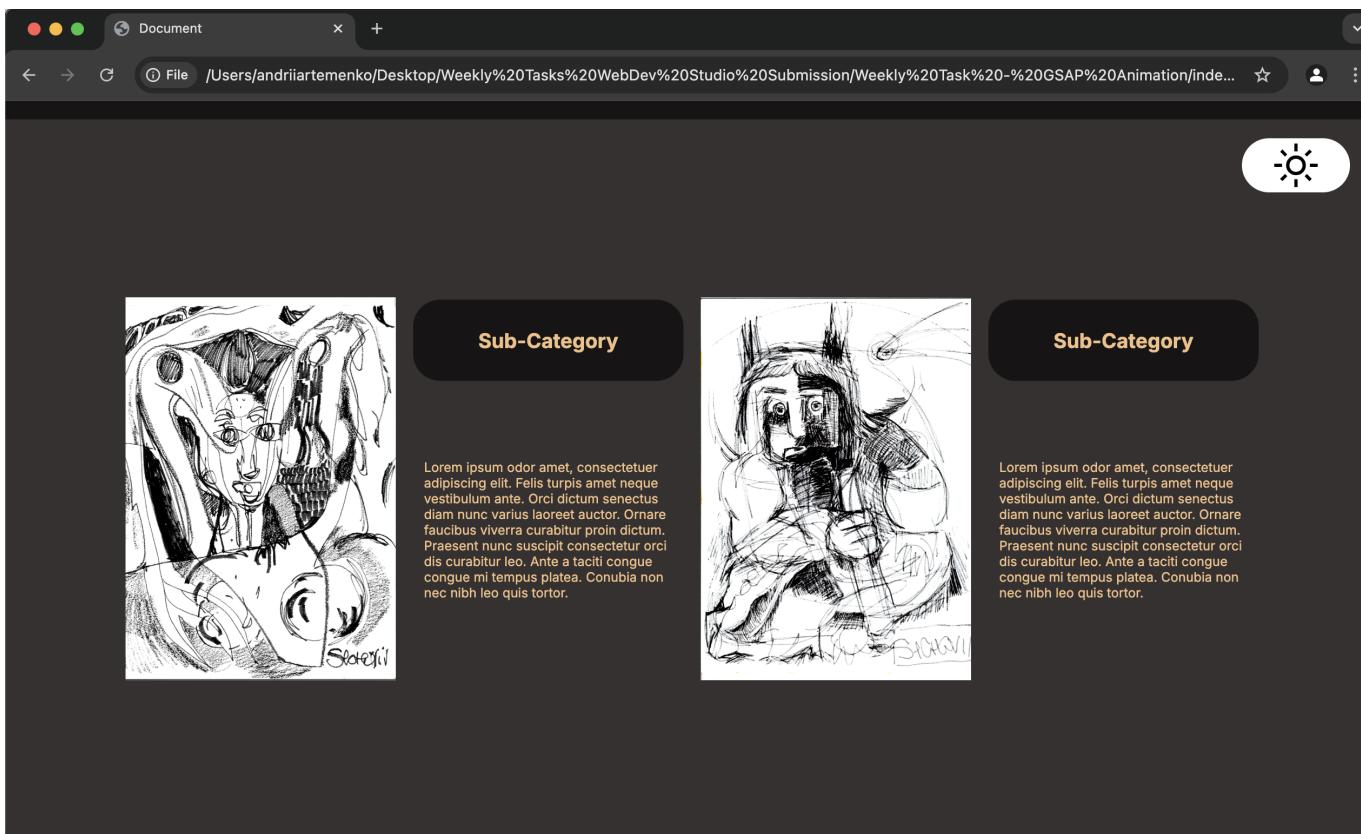
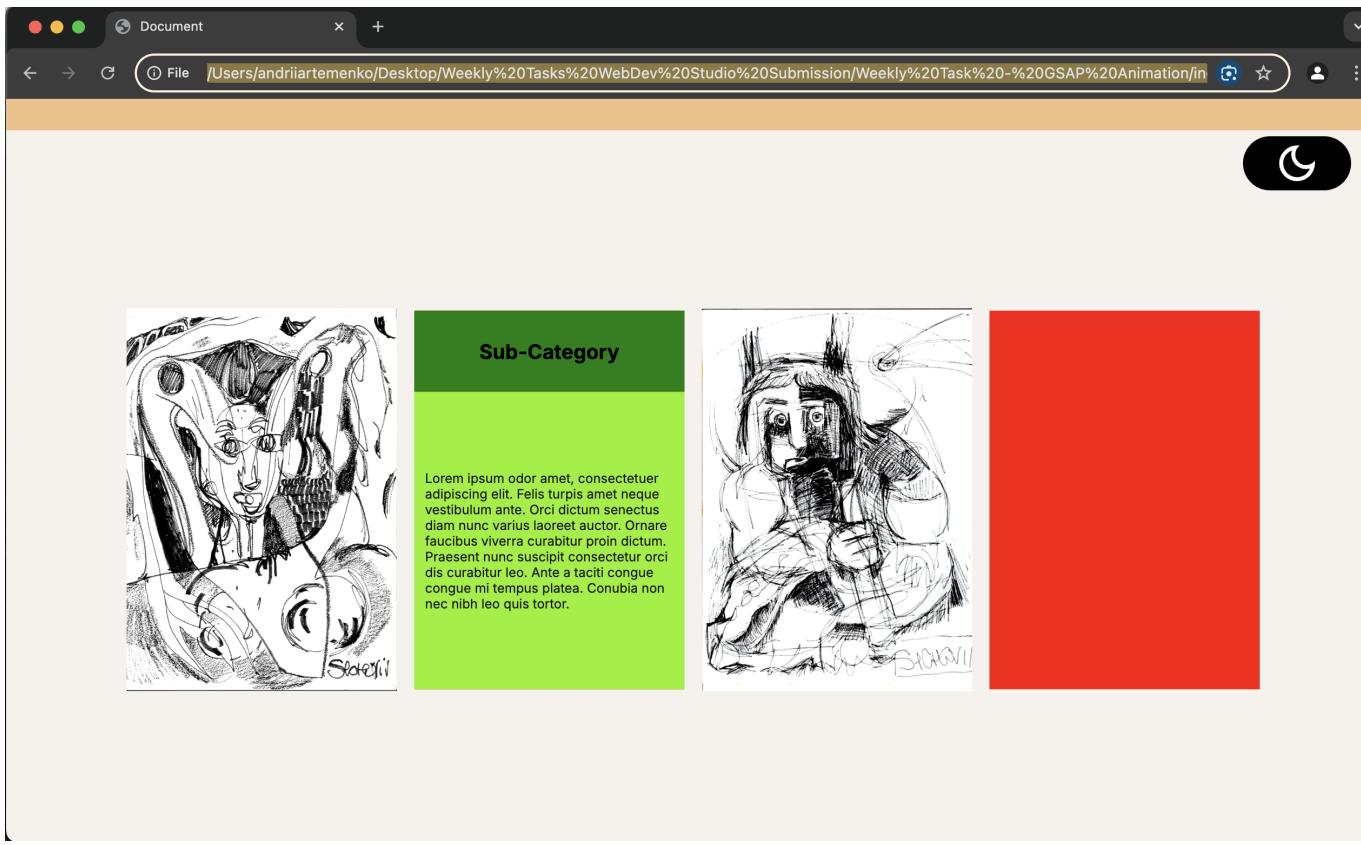
The screenshot shows a web browser window with a dark mode sidebar on the left. The sidebar contains icons for file operations, search, and other tools. The main content area displays a CSS file in a code editor.

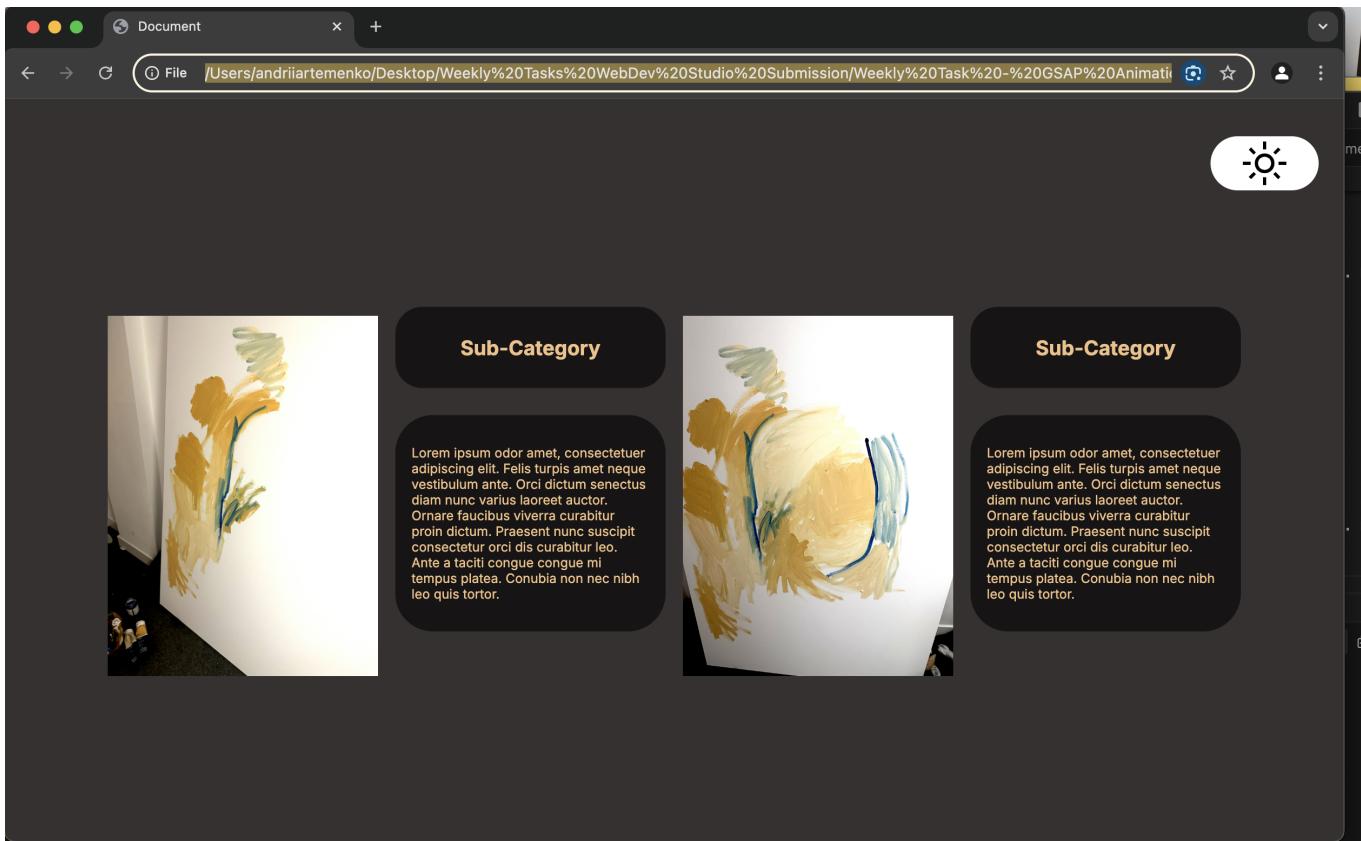
The CSS code is as follows:

```
# style.css > .test
50 .Pic {
51   position: absolute;
52   object-fit: contain;
53   column-count: 4;
54 }
55
56 .Pic img{
57   object-fit: contain; /* important to display nicely */
58   height: 500px;
59   width: 250px;
60 }
61
62 .test {
63   height: 400px;
64   width: 250px;
65   background-color: red;
66 }
67
68 .Intro_H1 {
69   color: var(--primary-text);
70   font-size: 2.25rem;
71   font-weight: bold;
72   display: flex;
```

The code editor has tabs for index.html, Slide2\_element2.jpeg, dark\_mode.svg, style.css (which is currently selected), Slide2\_element1.jpg, and app.js. Below the code editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE (which is selected), TERMINAL, and PORTS. A filter bar at the bottom right says "Filter (e.g. text, !excl...)"







I have then imported GSAP library, specifically scroll trigger for the main part, and built a simple js scrolltrigger effect, that pins the section to the top and only scrolls down after a certain period of time. GSAP has extensive documentation so it has been really easy to build this gsap scroll animation. however, when I tried to organise it into a timeline, to run the text on lside 1 drop animation, it would disable even the working of other javascript code that is needed for theme switch, so I decided to just do the scroll trigger effect ( ScrollTrigger, n.d.)

```
document.addEventListener("DOMContentLoaded", (event) => {
  gsap.registerPlugin(ScrollTrigger, RoughEase, CustomEase)
  let sections = document.querySelectorAll("section");
  sections.forEach((section) => {
    ScrollTrigger.create({
      trigger: section,
      start: "top-top",
      end: "bottom-top",
      pin: true,
      anticipatePin: 1,
    });
  });
});

let darkmode = localStorage.getItem('darkmode')
const themeSwitch = document.getElementById('theme-switch')

const enableDarkmode = () => {
  document.body.classList.add('darkmode')
  localStorage.setItem('darkmode', 'active')
}
```

## Lift of References:

1. Coding2GO (2024) 'Create a Dark Mode Switch with HTML, CSS, JavaScript', *YouTube video*, 5 August. Available at: [https://www.youtube.com/watch?v=\\_gKEUYarehE](https://www.youtube.com/watch?v=_gKEUYarehE) (Accessed: 4 February 2025)
2. Greensock. (n.d.). *ScrollTrigger*. Retrieved from <https://greensock.com/docs/v3/Plugins/ScrollTrigger>