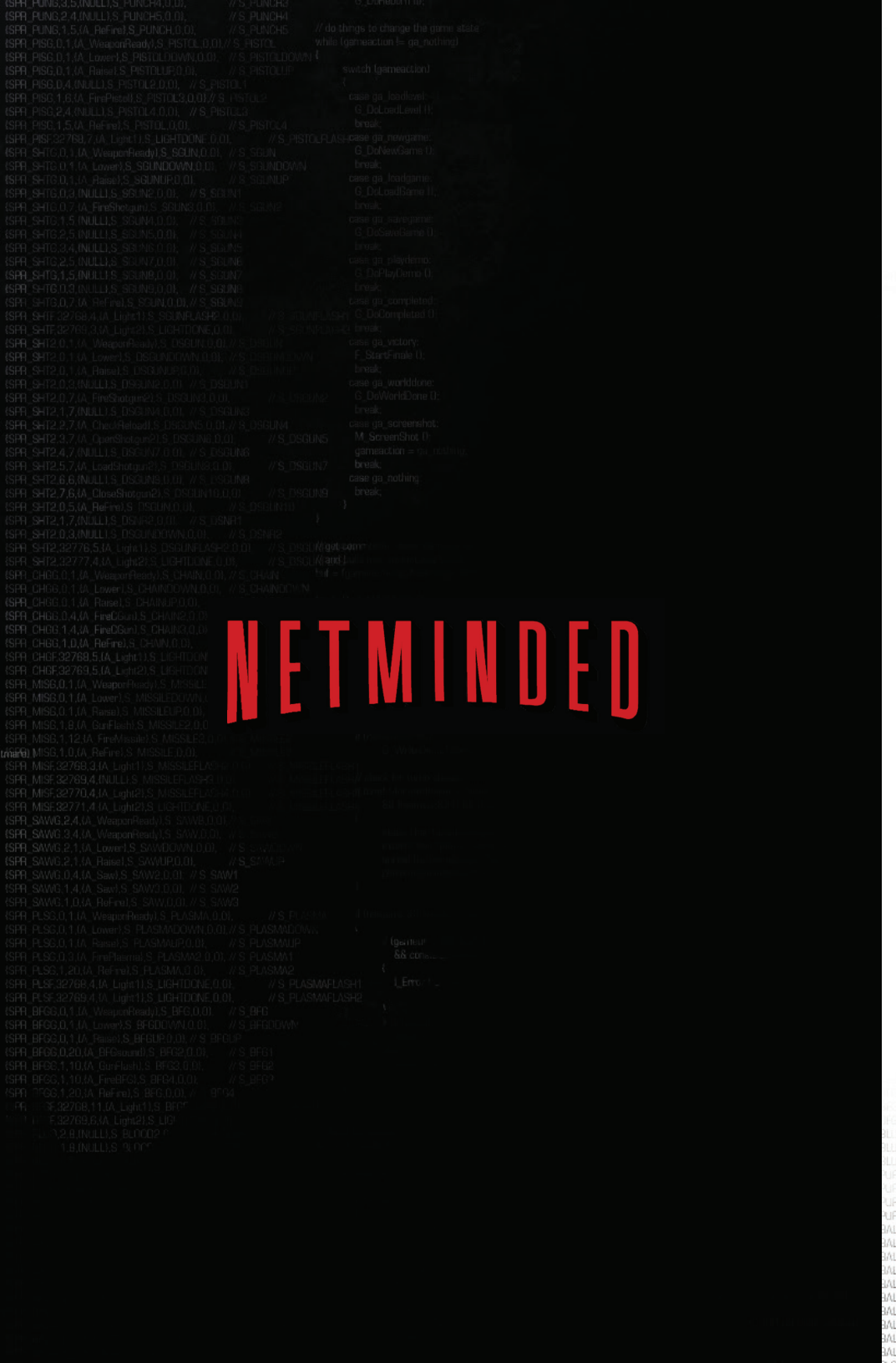


EVIL AI

Vsevolod Ivanov
Marie Pontais
Daniel Munoz

INTRODUCTION

Netminded is a project that started based on the question: how can we create an interactive website that reflects a certain personality on its actions, responses and interaction with its user?

By designing an “artificial intelligence” program that imposes its mandates over the user, we attempted to recreate an “evil” code that regardless the input of the user, it would take it’s own decisions, use its own methods and finally impose its own personality.

This represents a critique of the algorithms which drives the personalization of the software for its end user. Often, on many social platforms as Instagram, a person has a false impression of empowerment over its content whereas in reality the content of his feed is dynamic adjusted according to the backend algorithms. This “filtering” is performed implicitly and it is not always accurate nor it has a positive impact for the user.

The Netminded project exposes this neverseen programmed behavior by explicitly judging the human in the front of the computer. It achieves this by using the video processing captured from the webcam which it uses to judge the user according to its colors.

Netminded is programed to be evil, judgmental, stubborn and proud. It is a reflection of our social networks on the net and their behaviours, it implies how everyday we are exposed to so much filtered information that only reafirms our decicions, by no allowing us to reicive other opinions and perspectives.

The website was contructed inspired in the webpage “Netflix”. We chose this site because it allows simple interaction and offers a variety of options, that later on can be filtered, also it allows us to play with the contents (the movies) in a more dynamic way like creating new movies that reflect the personality of the AI and create generes that alo reflect the personality of the user.

WORK PROCESS

The first step of the project was to decide what existent webpage we would emulate, and we would proceed to implement our own version using the AI.

After that we started listing the functions the AI would implement in the interface, how it interfere with the normal use of the the webpage and how it would represent accurately its stubborn personality, for this we decided to implement xx different functions:

1. As soon as the user loads the landing page and tries to go further by clicking the "get started" bottom, the AI would ask for permission to activate the camera, not letting the user to advance any further if the camera is not activated.

2. After that the user is redirected to the second page were the three carrousel of movies are located, over here the camera is taking record of the colors in front of the screen, by selecting the most prominent color the AI displays a specific category of movies that matches the color.

3. The user then is only allowed to chose a movie from such specific category, hiding all the other genres.

4. Several movies are displayed but only a small amount of movies are actually linked to a third page. This movies are fake movies created to symbolize the personality of the AI. All the rest movies are locked and coded so if the user tries to access them the movie image will shake three times before redirecting the user to the initial page.

5. A sound library will be annexed to the document script in order to let the user know that the movies selected are not valid options for the AI

6. Finally when the user agrees to click on one of the AI movies it will be redirected to the "movie" recreated using another library for video streaming on the background of the page.

After having each function defined we split the workload in three main different branches, interface structure, camera manipulation and population of carrousels. These would be main structure of our work and then the three processes would be merged into the main page.

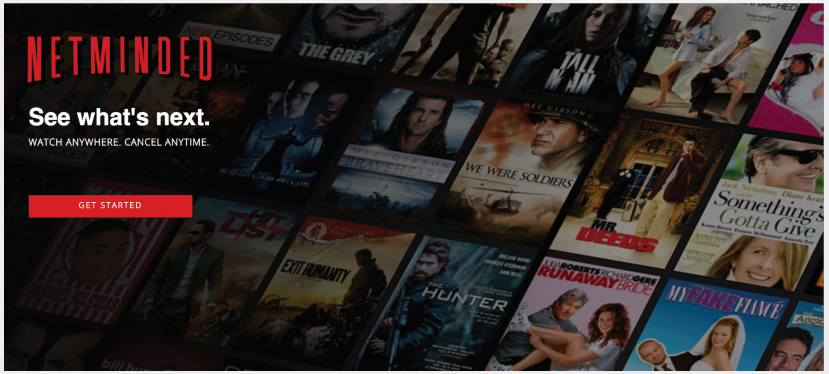
TECHNICAL PROCESS

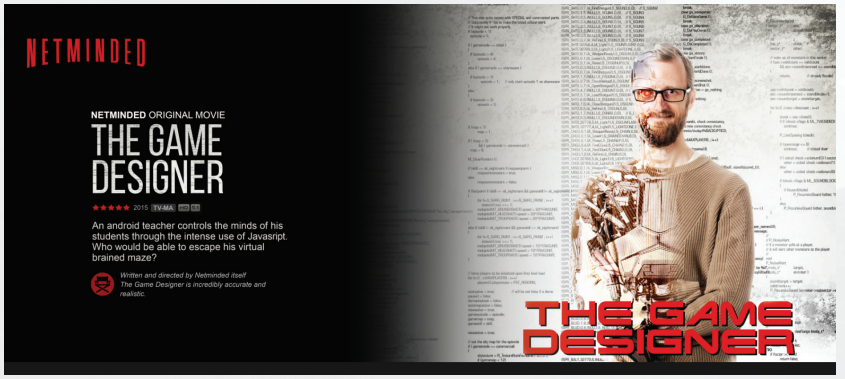
Aesthetics & Structure

The structure of the web page is fairly simple, There is landing page using a container that uses a background image to emulate the Netflix background. The background is styled in CSS to expand when the mouse hovers over the container.

In the landing page the camera is located in the background of the page so the user cannot see the images being reproduced back at him/her. And right under the logo the is the main link to be redirected to the second page. The <html> file also contains the source code for the four main libraries implemented in the document, as well as a second container for the camera.

The second page contains a second version of the logo, as well as an example banner with one of the AI movies across the body Right underneath there are three carrousels containing the different arrays of the movies that are called once the camera has identified the colors in front of the screen. Each carrousel has styled arrows to navigate horizontally, and finally there is a footer that contains a “contact link” and a “Copyright trademark”.



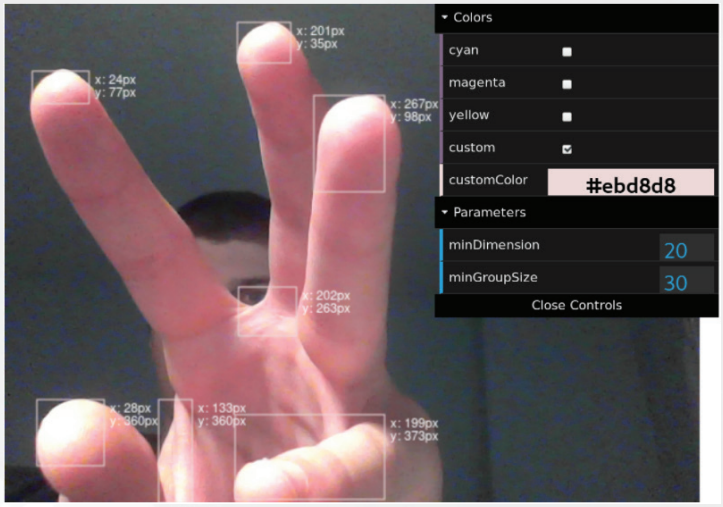


TECHNICAL PROCESS

Video Processing

Finding the adequate video library on the web was not easy, it was rather complicated, considering that we needed a real-time video processing library using only the web-browser power. The thing is that most of the real-time processing is done in low level programming languages as C++ to optimize the speed for this high in-demand task. We performed some research that lead me to OpenCV which is a library of programming functions mainly aimed at real-time computer vision. Even if there are binding to this library in a more high level language as Python, we still needed it for the web. Then, we could have used Node.js OpenCV bindings to run a server to deliver the video processing results. However, the idea of running the it directly in Javascript seemed to be attractive. In the same time it would represent a challenge to push the web to its limits.

This challenge led us to discover Tracking.js, a Computer Vision on the web. It looked very promising and in fact it works so efficiently that we had incredible results. However, the arrival wasn't so convincing when none of the examples worked. By chance, it was a small error in the referenced libraries for the demos controls. Seva happily contributed by fixing it.



Tracking.js

The examples were using a library that allows to interact with the color variables to alter the parameters of the video capture.

You can play with it at :
https://sevaivanov.github.io/netminded

One challenging aspect was calibrate the colors. In fact, Tracking.js detects precisely shapes of certain minimum dimensions. However, if you go lower than 10 as minDimension value than the detection is very poor. Also, the colors have gradients which would also influence the speed because many tracking events call would happen for the same color with different gradients.

TECHNICAL PROCESS

Merging

Tracking.js reads from the video html tag and it can write to the canvas. We need it as a hidden from the user feature because. After all, it is an evil AI that is supposed to misbehave. Turning the display off the div container or individually its content didn't worked because the tracking wasn't occurring. The solution was the z-index that allows elements to superpose one on top of another. We assigned a higher z-index to the main page container and it overlayed my video & canvas tag that we decided to keep there for debugging purposes. Actually, you only need to turn off the z-index variable in the browser inspector to see what is really happening during the tracking which is quite handy.


```
// The colors we want to track
var cameraColors = {
    pink:
    {
        hex: '#877f7f',
        rectCounter: 0,
        percentage: 0
    },
    red: {
        hex: '#ffffff',
        rectCounter: 0,
        percentage: 0
    },
```

Logic

Seva started by creating an object to hold color-relevant data. From there he set the tracker to track the video tag and use the tracker on it.

tracking.track('#video', tracker, { camera: true });

The he listened on the tracker for a track event that would trigger a function where he would do all of the Color logic processing. On this event, you receive data in a form of callbacks for each rectangular frame called rect in its context.
To time the tracking to happen only every 5 seconds he used capInterval to which he added a desired delay of 0.1s after which he would stop the capturing to give some time interval to process the color logic so that the movie logic situated in the Marie's P5 code can be reload them in the webpage. This interval is needed to allow the user to notice the change and have the time to see the suggestions if he moves a lot.

TECHNICAL PROCESS

Logic

```
// Capture vars to time the color processing
var capStart = (new Date()).getTime(),  // secs
    capInterval = 5,                    // secs
    capDelay = 0.1;                     // secs

tracker.on('track', function(event)
{
    cameraReady = true;
    context.clearRect(0, 0, canvas.width, canvas.height);

    event.data.forEach(function(rect)
    {
        // Colors logic
        //console.log(rect.color);

        // New capture - reset parms
        if (diffInSeconds(capStart) == capInterval)
        {
            for (var name in cameraColors)
            {
                cameraColors[name].rectCounter = 0;
                cameraColors[name].percentage = 0;
            }
        }
        // It will look every n sec (interval) during k sec (delay)
        if ((diffInSeconds(capStart) >= capInterval) &&
            (diffInSeconds(capStart) <= (capInterval + capDelay)))
        {
            //console.log('timed color capturing ');
            /* FIXME:
             *  Getting called too many times even with 0 delay
             *  Maybe due to the tracking events going too fast
             */
            updateColorPercentage(rect);
        }
```

When the capture started, he would reset the previous capture arguments. After, he would update the color percentage for each rectangle captured seen below.

```
function updateColorPercentage(rect)
{
    var name = rect.color,
        canvasSurface = canvas.width * canvas.height,
        rectSurface = rect.width * rect.height;

    var rectSurfacePercentage = Math.floor(rectSurface / canvasSurface);

    var newPercentage = cameraColors[name].percentage + rectSurfacePercentage;

    if (newPercentage > 100)
    {
        newPercentage = 100;
    }

    cameraColors[name].rectCounter++;
    cameraColors[name].percentage = newPercentage;
}
```

When the capture started, he would reset the previous capture arguments. After, he would update the color percentage for each rectangle captured seen below.

As seen previously, I saved the number of rectangles to be able to debug one error. Unfortunately, it seems that even with a 0.0000000000001s delay I would get at least 10 rectangles. There could be many reasons to that but I was short on time to resolve it. In fact, we had to merge the P5.js and normal Javascript code together. Plus, we had Jquery at few places. It definitely takes time to merge different ways of coding.

TECHNICAL PROCESS

Logic

Due to initial camera detection delay, we are initializing the movies to random before any detection to allow the user to see them directly.

```
67      <script>
68          $(document).ready(function()
69          {
70              initCameraTracking();
71
72              // No camera at start = random colors to initMovies
73              var randomColors = {};
74              for (var name in cameraColors)
75              {
76                  randomColors[name] = {
77                      percentage: getRandomInt(0, 70)
78                  };
79              }
80              initMovies(randomColors);
81          });
82      </script>
83  </head>
```

Afterwards, we are using the global camera variables to reload the movies with a 10 seconds delay. The camera detection and the movies reloading is not timed because the P5 usage requires an interaction in the main loop whereas the tracking is happening in an event. Therefore, to make sure to eliminate the reloading of the movies to be mixed from two captures we times the capture at 5s and the reloading at 10s to have a little movie reloading delay of 5s. This would allow our project's user to have a good experience even with a slower network connection.

Finally, the only thing that is less configurable is the colors calibrations using the variety of tones. Right now, we have to hardcode the values. It could be interesting to have a panel admit but in the same time if this was a real thing they would be already preloaded using all of the tones expressed in the hexadecimal numbers.

```
92 var notSpeakYet = true;
93
94 function draw()
95 {
96     updateMoviesInterval = int(millis() / 1000);    // in secs
97
98     if (cameraReady)
99     {
100         if (cameraHasColors())
101         {
102             console.log('cameraColors detected');
103             console.log(updateMoviesInterval);
104             if (updateMoviesInterval % 10 == 0)
105             {
106                 console.log(cameraColors);
107                 initMovies(cameraColors);
108             }
109         }
110
111         if (resetSlider)
112         {
113             displayMovieSliders();
114         }
115     }
116     else if (notSpeakYet)
117     {
118         responsiveVoice.speak("Is there anyone there?");
119         notSpeakYet = false;
120     }
```

TECHNICAL PROCESS

Movies Research

The research started with this project by selecting 150 movies, 30 for each type (girly/horror/documentaries/childish/action). For this part of the assignment, Marie mainly used Imdb and browsed movies according to their style, and picked (more or less randomly according to my own knowledge of these movies) 30 of them. First draft:

http://hybrid.concordia.ca/m_ponta/CART351/Ass3/

It implemented the arrays of all the different types, with their image and title. You could go through each array with the up arrow, and change type with the enter key. But here, nothing was trying to pick the movies according to given color values

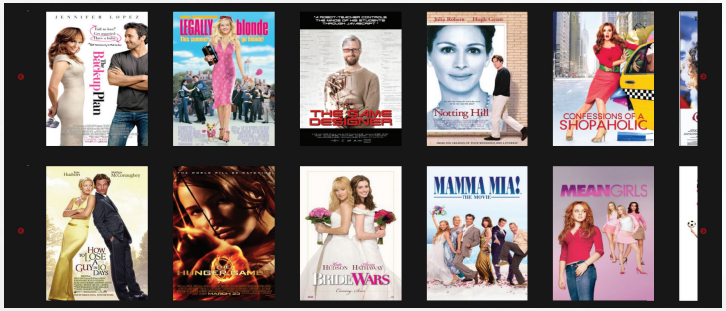
Marie implemented then the matching the of movies to the camera. She started implementing my checkValue() function. She had a really hard time implementing this function, as for many reasons, finding the logic behind creating a new array, picking different films in all the previous ones she made, was pretty challenging. Second draft:

http://hybrid.concordia.ca/m_ponta/CART351/Ass3/index3.html

Finally she ended up fixing after a few hours of re-thinking and re-coding my logic, adding function and calibration so that everything was working with a total of 30 picked movies by the A.I.

You can see this same simple functionality (up arrow to go through the films), but according to given color values functionning right here:

http://hybrid.concordia.ca/m_ponta/CART351/Ass3/index3.html



TECHNICAL PROCESS

Implementation of Movies within the Slider

This part was also one of the main challenge of this project for Marie. Because she did not make the carousel code herself, and given the numerous CSS and js files it had, she had a hard time understanding how the slick library worked.

Daniel slider's was working perfectly if the "center slider" class section was populated in the html file, but when she tried doing it dynamically, it ended up breaking everything.

There were basically 2 bugs, first, no slider would be on the website, and only three small img icon would appear saying that the slider section were empty.
As she discovered later, the slider were creating themselves only once, and at document ready. So she had to put the Jquery needed for the sliders into a custom function, which had to be called only when the images had been added. And magically this fixed it!

Second, she added a spacebar command, where each you would hit the space bar, it would randomize new colors, has if the camera were refreshing. The slider were displaying nicely the first time, but as she touched the space bar, it would display no more the slider, and all the picked movies would display themselves one after the other as you were scrolling through the page.

This bugg was caused by the fact that the slider made by the slick library, instantiate themselves dynamically too! So that once you had created your first slider, the class of the actual css section no longer "center slider", but "center slider initialized slick…….".
So the fixing to this new problem was to make a new function, which she called resetSlider(), which basically call each slider by its Id (slider1, …), make them empty, and resets their class name to "center slider". This way, she was able to reset these sliders each time she needed to change the movies picked: each time the camera was refreshing (well mimicking this with the random colors).

TECHNICAL PROCESS

Syncronizing Netminded & camera

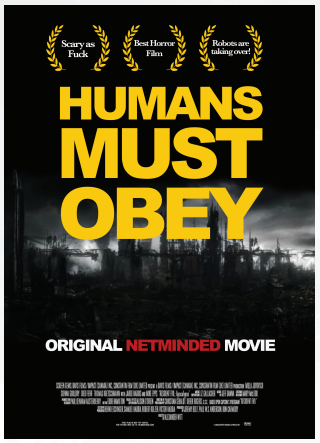
This part was the most challenging one. Because our codes ways of speaking to each other hadn't been clearly set from the beginning, the implementation of the both together was not working. And we had a hard time figuring out how to make them work together. So after a lot of hours going nowhere, Seva ended up recoding a bit of the sketch.js file and made everything work together so that we could present today.

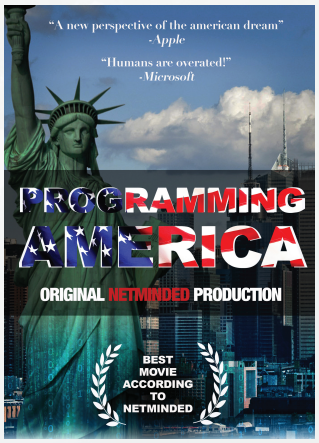
The last part of this process, was to make the personality of the A.I (Judgemental, Stubborn, and Proud) be revealed through the UI. For this, I first implemented a Jquery effect, making each "badMovie" you select shake. On top of that, the badMovieCount stores how many time one chose the human movies over the A.I, and make Netminded say different things accordingly (here I am using the ResponsiveVoice library). Finally, implementing two animations (one in P5, the other based on the beach party example), so that if you try too many times to select badMovies, the A.I turns off your screen (P5 animation).

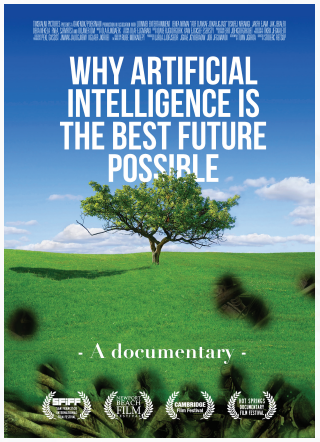
Otherwise, when you select the A.I original films, than it sets a new page to be displayed, which plays a part of the real Animatrix youtube video, with its sound, explaining how machines took over the world.

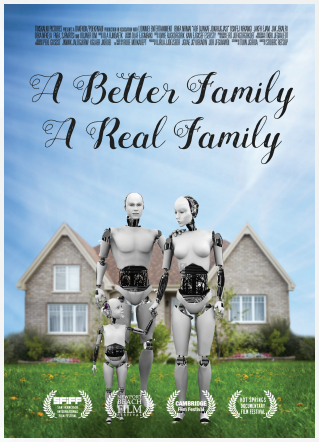
TECHNICAL PROCESS

In order to make a more consistent imagery we also designed six different movie posters so the user was able to chose movies that the AI had directed itself. This movies had dystopian themes related to the AI personality and all of them depicted technology and artificial intelligence as the main topics:









NETMINDED