Team No. 11

Play With Images

https://github.com/jaffreyjoy/tinker-pixel

Introduction:

We have developed an image-processing web application that accepts an image as input into the canvas of our front end and transforms it into the desired processed image. This web application can convert an image into 18 various formats.

Technology Stack:

Font end:

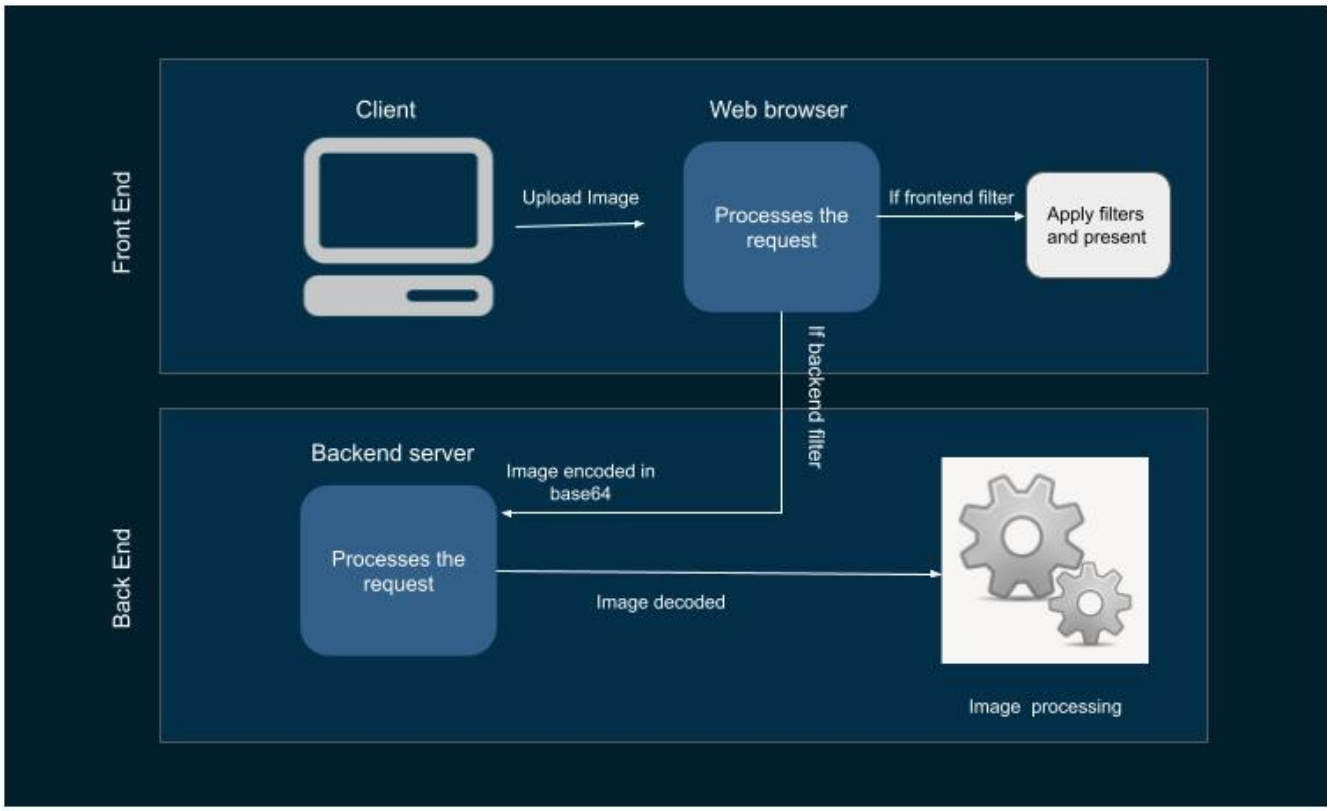
- React JS framework
- React UI binding layer-Redux

Back end:

- Fast API framework
- OpenCV
- Haar cascade object detection library

Theory:

Architecture:



The client uploads an image for processing on the front end. Then it chooses the filters it wants to use. After accepting the image, the browser separates it into a "frontend filter" and a "backend filter" depending on the filter that was selected. The "frontend filter" is dealt with at the front end and processed there with very little latency, whereas

the "backend filter" requires that the image be encoded in "base64" format and sent to the backend server where it is decoded back into an image stored in a NumPy array in RGB bit format. Then, additional processing using functions on this array is carried out.

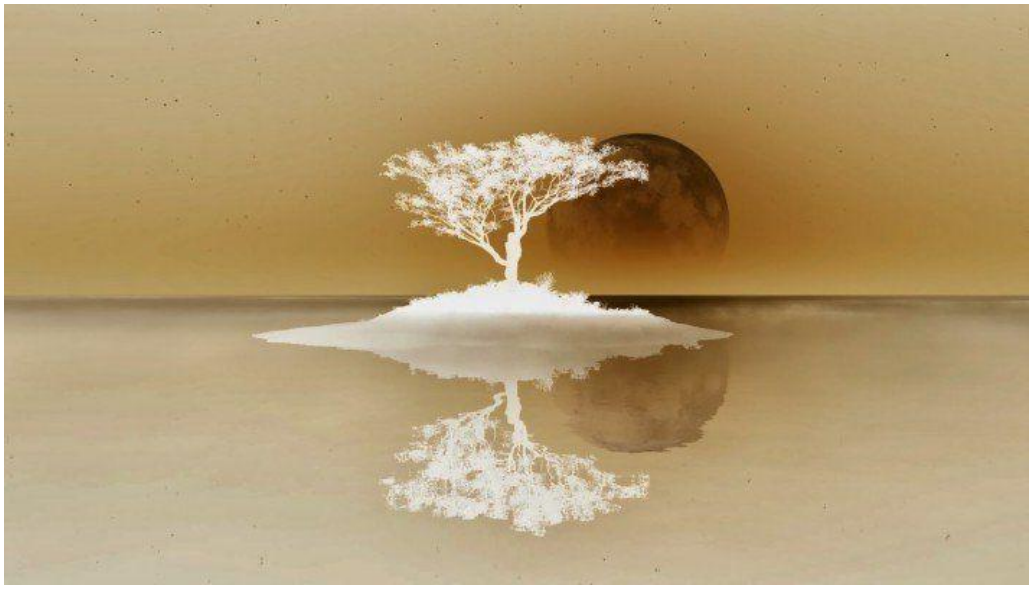
1. **Front-End dealt filters:**
   In order to apply these filters, let's consider the following sample image:

The following are some of the front-end filters applied

- Grayscale:

Grayscale is a spectrum of apparent-colorless hues of grey. Black, which is the complete absence of transmitted or reflected light, is the darkest tint imaginable. White, which is the complete transmission or reflection of light at all visible wavelengths, is the lightest tint imaginable.

- Invert:



Invert is reversing the color of an image. When referring to an image, this is often called a negative.

- Hue Rotate:

Hue describes the color as it appears on a grayscale, color wheel, or red-green-blue scale. Brightness increases cause the primary colors or grey presented to appear more intense, but the color itself remains unchanged. There is no difference when color values are changed.

The other front-end filters are

- Brightness
- Contrast
- Opacity
- Blurr
- Saturate
- Sepia

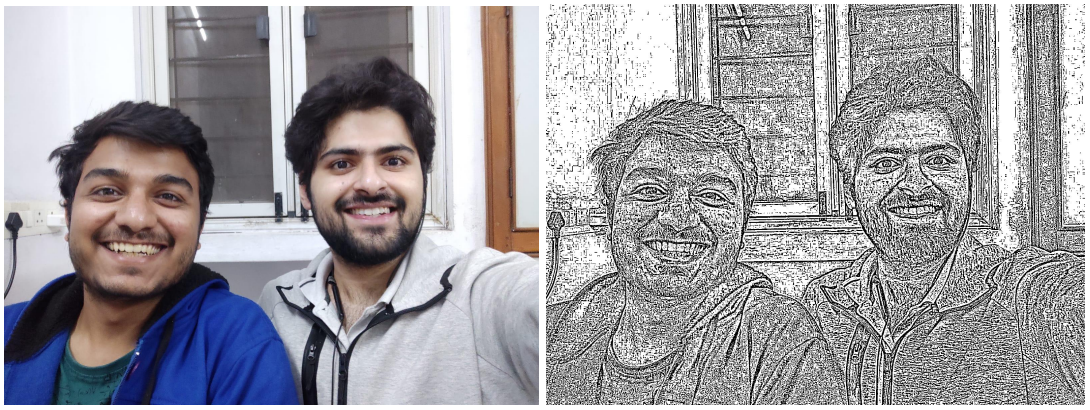
2. **Back-End Filters**

   The following are some of the back-end filters

- Specs-ify

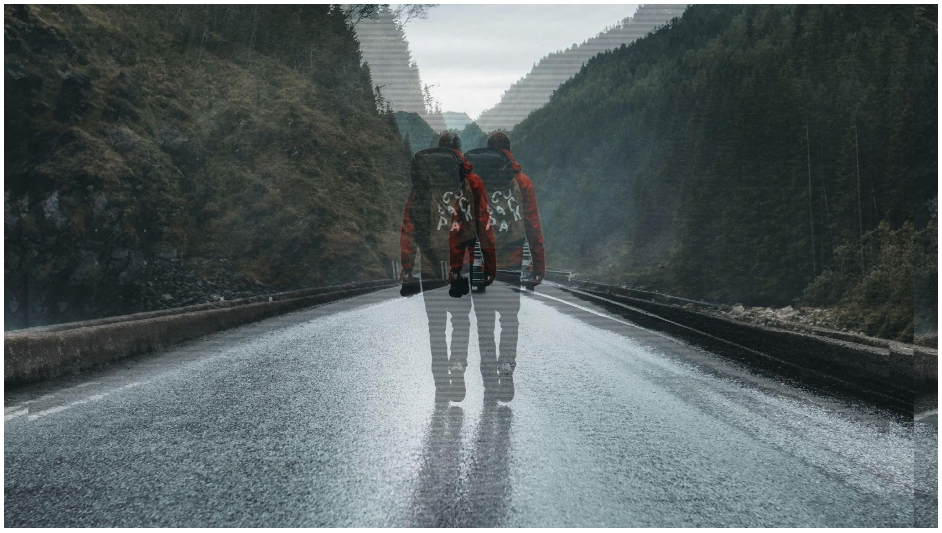
We have used the Haar cascade object detection library in python to detect the eyes on the faces of the people and then apply spectacles.
It is best suited for images having faces.

- Experts-pencil



Drawing with a graphite-based tool housed in a wooden case is called a pencil drawing. It might be a finished piece of art, an exercise in visual expression, or a sketch for a more elaborate piece in another medium. This expert sketch provides such an effect with better detailing and heavy concentration.
Best suited for images requiring greater sketch sharpness.

- Lost in the world



This provides us with the visualization of a horizontally distorted image, also producing a shadow effect to the image.
The other back-end filters are

- Cartoonify
- Colors of Life
- Blurred beauty
- Oil painting
- Thanos Effect
- Pencil sketch
- Sketch the scene

Implementation:

Architectural implementation is already explained in the above sections. The following is the implementation of some backend filters:

- Speci-fy
  Speci-fy filter puts specs on the faces. It is designed to look for all the eyes in the image frame using a cascade classifier and then it saves the coordinates of the eye in a list(It is basically a tuple of starting x,y coordinate width, and height). After all the eyes have been classified then it will take those coordinates, width, and height and then plot the spec sticker on the eyes after resizing the spec image for that particular height and width.
- Lost in the world
  This filter divides the image into horizontal strips of length 1 pixel and then alternatively shifts them in the opposite direction to create a shadow-like effect on the picture.
- Thanos effect
  This filter works similarly to the lost in world filter but the image is instead shifted in random directions

Limitations:

1. Haar cascade library is a very old library and comes with its limitation. When trying to detect eyes in the “specs-ify” feature, it sometimes can detect eyes in an image containing no such eye object.
2. The back-end dealt filters have a higher processing latency in comparison with the front-end dealt.
3. PNG images are not supported for any of the images.

Future Scope:

1. Integration of certain filters such as seam carving, cartoonication, beard and hat stickers led to unexpected result while integrating with the application hence not included in the main app. However implementation, input/output of such unintegerated filters are saved in a separate folder with the code.