To approach this, I loaded in the static image and tried a couple different methods at first. I actually started by converting the image to grayscale to see if I could just pick out the shapes based on intensity, but that didn't really work since the shapes are defined more by color than brightness. After that I switched to HSV, which makes it way easier to define color ranges.

I set up a dictionary of the main colors I wanted to detect, built masks for each of those, and then combined them into one big mask. After that I did some morphological cleanup (close + open) just to get rid of the random noise and smooth out the regions.

Once I had a clean mask, I ran contour detection to pull out all the possible shapes. I filtered out anything tiny since those are usually just noise. For each contour that made it through, I approximated it to simplify the edges, drew an outline around it, and marked the center point so I could actually see where the program thought the shape was. Then I did a quick check on the number of vertices to guess the type of shape (triangle, square, circle, etc.).

On top of that, I also grabbed the average HSV values inside each contour so I could label the color of the shape. That part was a little trickier, since sometimes the ranges overlap or the lighting can throw things off, but it mostly worked.

The main challenges I hit were with the color detection – gradients and background colors can bleed into the mask if the ranges aren't set carefully, so it took some trial and error to tune them. Also, approximating circles vs polygons is never perfect, since OpenCV just sees a bunch of vertices. But overall, the pipeline of mask → contours → filter → label worked pretty well.