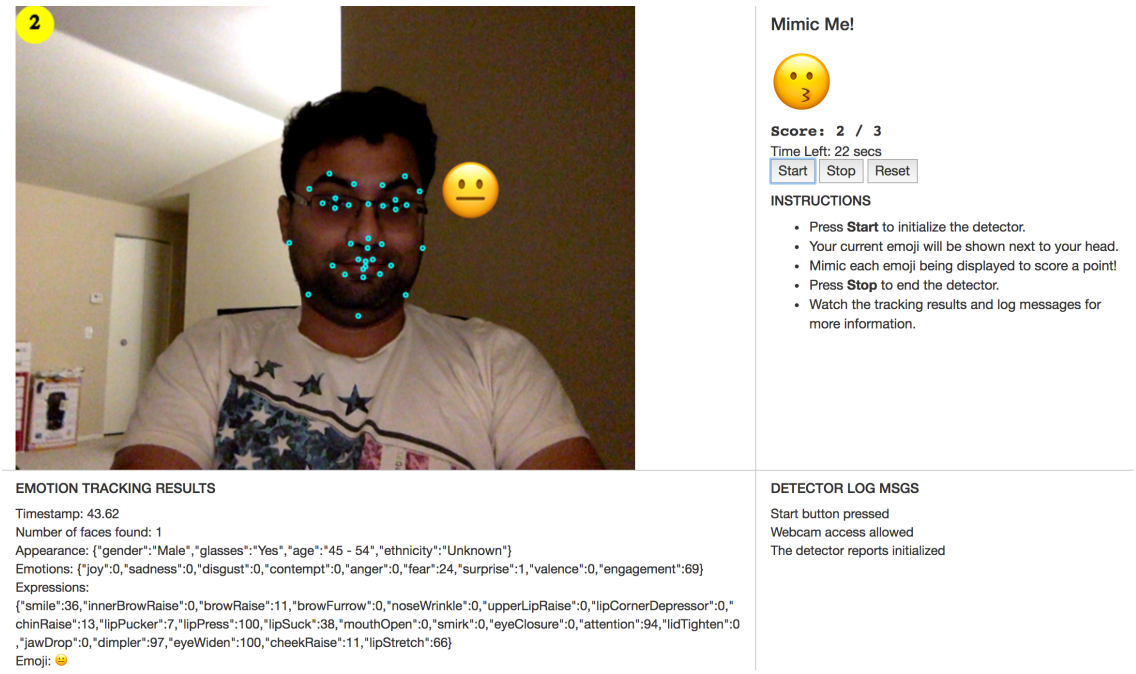
12/2/17

PROJECT MIMIC – CV CONCENTRATION

1. DASHBOARD



The mimic game uses a set of simple emojis (easily recognized ones) and a score of 14 was achievable within 60 seconds. The game is equipped with a timer function that stops the game when time is up.

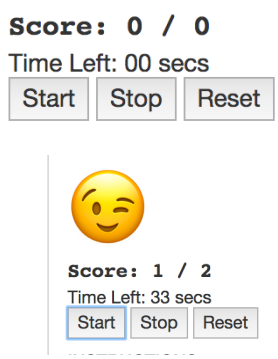
2. IMPLEMENTATIONS - OVERVIEW

The following set of functions were implemented

A. Mimic.js

i. setTime : A custom function that sets the timer on the screen. Sets the game for 60 seconds.

```
// Display Timeleft counter : ADDED FUNCTION
function setTime(sec){
    $("#timeleft").html("Time Left: " + sec + " secs");
}
```



ii. onStop : Added a line to clear the timer, if running.

```
// Stop button
function onStop() {
  log('#logs', "Stop button pressed");
  if (detector && detector.isRunning) {
    detector.removeEventListener();
    detector.stop();  // stop detector
  }
    if(timerSession){
        clearInterval(timerSession);
    }
};
```

s

iii. onReset : The reset button is added with a customization to clear the timer.

```
// Stop button
function onStop() {
  log('#logs', "Stop button pressed");
  if (detector && detector.isRunning) {
    detector.removeEventListener();
    detector.stop();  // stop detector
  }
    if(timerSession){
        clearInterval(timerSession);
    }
};
```

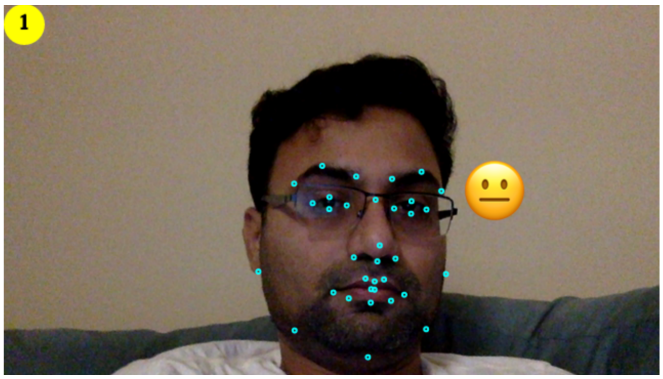
iv. drawFeaturePoints : Draws set of featurepoints in cyan color.

```
// Draw the detected facial feature points on the image
function drawFeaturePoints(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

  // TODO: Set the stroke and/or fill style you want for each feature point marker
  // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D#Fill_and_stroke_styles
  // <your code here>
  ctx.strokeStyle = 'cyan'//
  ctx.lineWidth = 2;

  // Loop over each feature point in the face
  for (var id in face.featurePoints) {
    var featurePoint = face.featurePoints[id];

    // TODO: Draw feature point, e.g. as a circle using ctx.arc()
    // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/arc
    // <your code here>
    ctx.beginPath();
    ctx.arc(featurePoint['x'],featurePoint['y'],2,0,2 * Math.PI,true);
    ctx.stroke();
//    ctx.fillText(id, featurePoint['x'],featurePoint['y']); // to find the index of left point to position the emoji
  }
}
```



v. drawEmoji : Places the dominating emoji at the left top corner of the face.

```
// Draw the dominant emoji on the image
function drawEmoji(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

  // TODO: Set the font and style you want for the emoji
  // <your code here>
    ctx.textBaseline ="hanging";
    ctx.strokeStyle = 'red';
    ctx.font = '64px serif';
  // TODO: Draw it using ctx.strokeText() or fillText()
  // See: https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fillText
  // TIP: Pick a particular feature point as an anchor so that the emoji sticks to your face
  // <your code here>
    point = face.featurePoints[10]; // stick the emoji to the left of the face
    ctx.fillText(face.emojis.dominantEmoji, point['x']+20, point['y']-20);
}
```

vi. stopTimer: A custom function to stop & clear the timer on stop, reset button actions.

```
//Custome function that stops timer
function stopTimer(val) {
    clearInterval(val);
    setTime('00');
}
```

vii. initializeGame: A custom function to initialize the function.

```
// Custom function that initiates the game
// called while init and reset
function initializeGame(isReset){
    // initialize the variables
    resetVars();
    // console.log(emoji_ss);
    // start setting the target emoji in timely interval
    setTarget(cur_emoji);
    setScore(correct, total);

    // Run the timer only when it is started
    if (!isReset){
        var timer = 60; // set the game to 1 minute
        display = document.querySelector('#timeleft'); // 'timeleft' is the id added to HTML file
        timerSession = setInterval(function () {
            minutes = parseInt(timer / 60, 10)
            seconds = parseInt(timer % 60, 10);

            minutes = minutes < 10 ? "0" + minutes : minutes;
            seconds = seconds < 10 ? "0" + seconds : seconds;
            // Sets the time display at the html
            display.textContent = "Time Left: " + seconds +" secs";

            if (--timer < 0) {
                stopTimer(timerSession);
                alert("GAME OVER. You scored:"+correct+". Press Reset to play again");
            }
            }, 1000);
    }
}
```

This function also brings a alert message with the final score, as below.

viii. setTarget: A custom wrapper on setTargetEmoji to facilitate tracking the current emoji Unicode.

```
// Custom function to setting target emoji
function setTarget(code) {
    setTargetEmoji(code);
}
```

ix. resetVars : A supporting function to reset the global variables associated with the game.

```
// Custom function to init/reset the variables
function resetVars(){
    this.reset;
    emoji_idx=0;
    cur_emoji=emoji_ss[emoji_idx];
    total=1;
    correct=0;
    if(timerSession) {
        stopTimer(timerSession);
    }
}
```

x. playMatch: The core function that matches the dominating emoji with the target emoji.

```
// Custom function that plays the game of capturing mimic
function playMatch(canvas, img, face){
    // get the canvas context
    var ctx = canvas.getContext('2d');

    // check face match
    if (toUnicode(face.emojis.dominantEmoji) == cur_emoji) {
        console.log("Hurrey, Match");
        drawScore(canvas,'green');
        correct++;
        setScore(correct,total);
        if (emoji_idx < 5){
            emoji_idx++;
        } else {
            emoji_idx = 0;
        }
        cur_emoji = emoji_ss[emoji_idx];
        setTarget(cur_emoji);
        total++;
        setScore(correct, total);
    }
}
```

xi. drawScore: A custom function that visually updates the user with a 'green' flash when there is a match and keeps the score inside a circle at top left corner.


```
// Custom function that draws the score & visually indicates when the mimic
// Also draws the current score on top of the circle drawn
function drawScore(canvas, color){
    // Obtain a 2D context object to draw on the canvas
    var ctx = canvas.getContext('2d');
    // draw the circle (banner)
    ctx.fillStyle = color;
    ctx.beginPath();
    ctx.arc(20,20,20,0,2 * Math.PI,false);
    ctx.fill();
    // draw the Score on top of the banner
    ctx.strokeStyle = 'black';
    ctx.font = '20px serif';
    ctx.strokeText(correct,15,10);
}
```

B. index.html

i. #timeleft

```
    <div id="timeleft">Time Left: 00 secs</div>
```

The html file is updated with this additional id that captures the time left during the mimic game.

A set of global variables were created for manage the game as follows. The same were updated during initialize, reset and during the game.

```
// TODO: Define any variables and functions to implement the Mim
// Global variables
var emoji_ss = [128563, 128521, 128535, 9786, 128561, 128542]; /
var emoji_idx = 0;
var cur_emoji = emoji_ss[emoji_idx];
var total = 1;
var correct = 0;
var timerSession;
```