Mimic Me

This section of the report describes the implementation of the Mimic Me project using Affectiva's emotion detection API. The aim of this project was to implement code to draw points on detected facial co-ordinates returned from the API.

The project then required to implement a game mimicking the emoticon displayed. The rules of the game are simple, if the user successfully mimics the emoji, they receive a correct score. If the emoji is not mimicked within 10 seconds, they don't receive a score and the next random emoji is shown.

Facial feature points

The first step of this implementation was to identify the feature points from the face detected. The code below takes in a canvas, img and face when detected, it then uses a fillStyle and strokeStyle to colour the feature point. The loop then draws a radius of 1 circular point over each feature point.

```
// 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
  ctx.fillStyle = 'red';
  ctx.strokeStyle = 'red';

  // Loop over each feature point in the face
  for (var id in face.featurePoints) {

    // TODO: Draw feature point, e.g. as a circle using ctx.arc()
    var featurePoint = face.featurePoints[id];
    ctx.beginPath();
    ctx.arc(featurePoint.x, featurePoint.y, 1, 0, 2 * Math.PI);
    ctx.stroke();
  }
}
```

Draw Emoji

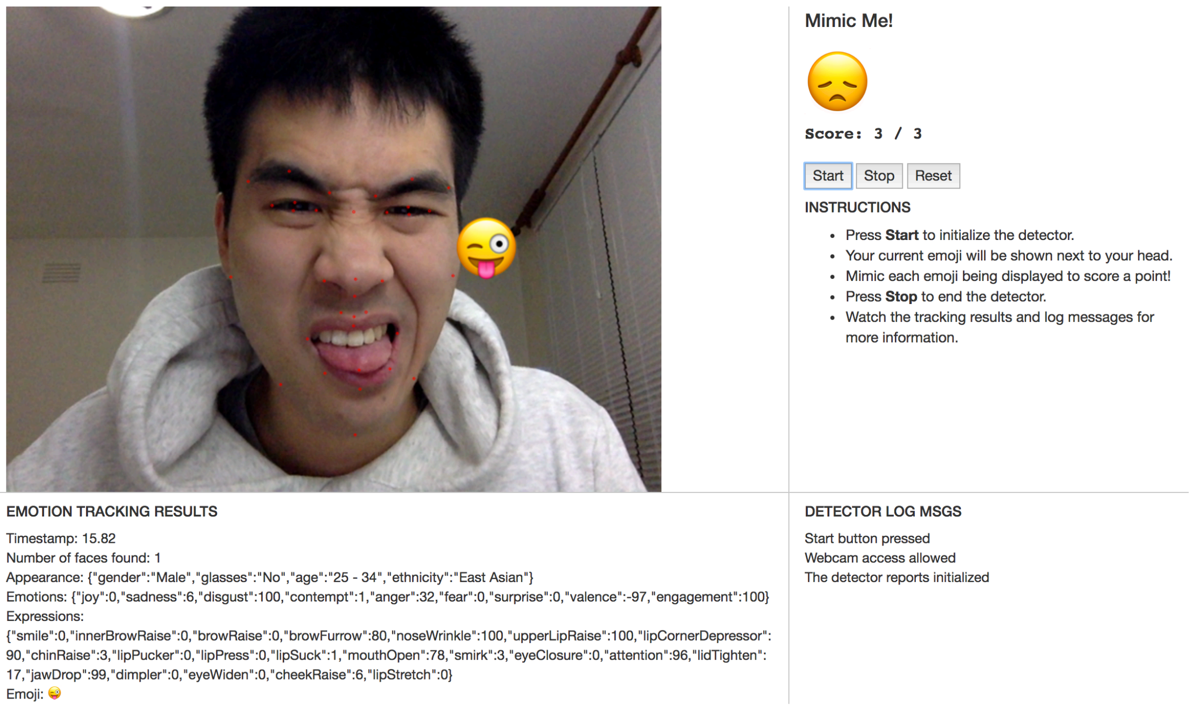
The next step was to display the dominant emoji which I've anchored to feature point 4.

```
// 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.font = '64px arial';

  // TODO: Draw it using ctx.strokeText() or fillText()
  ctx.fillText(face.emojis.dominantEmoji, face.featurePoints[4].x,
face.featurePoints[4].y);
}
```

Mimic Me Game



Mimic Me!

😞

Score: 3 / 3

INSTRUCTIONS

- Press **Start** to initialize the detector.
- Your current emoji will be shown next to your head.
- Mimic each emoji being displayed to score a point!
- Press **Stop** to end the detector.
- Watch the tracking results and log messages for more information.

EMOTION TRACKING RESULTS

```
Timestamp: 15.82
Number of faces found: 1
Appearance: {"gender":"Male","glasses":"No","age":"25 - 34","ethnicity":"East Asian"}
Emotions: {"joy":0,"sadness":6,"disgust":100,"contempt":1,"anger":32,"fear":0,"surprise":0,"valence":-97,"engagement":100}
Expressions:
{"smile":0,"innerBrowRaise":0,"browRaise":0,"browFurrow":80,"noseWrinkle":100,"upperLipRaise":100,"lipCornerDepressor":
90,"chinRaise":3,"lipPucker":0,"lipPress":0,"lipSuck":1,"mouthOpen":78,"smirk":3,"eyeClosure":0,"attention":96,"lidTighten":
17,"jawDrop":99,"dimpler":0,"eyeWiden":0,"cheekRaise":6,"lipStretch":0}
Emoji: 😜
```

DETECTOR LOG MSGS

```
Start button pressed
Webcam access allowed
The detector reports initialized
```

1. Generate a random emoji to mimic. This selects a random number from the length of the emoji array.

   ```
   // emoji randomiser
   function generateEmoji() {
       var emojiIndex = Math.floor(Math.random() * emojis.length);
       var emoji = emojis[emojiIndex];
       return emoji;
   }
   ```

2. The next step the code does is to check the detectEmoji is null, if so reset the game. This will ensure there will not be any errors with the detected emoji. The code then checks if the dominant emoji is detected or the start time is not null, if these are both valid, the game starts. When a successful match occurs, the code updates the score.

   ```
   // function to operate the game
   function playGame(detectedEmoji) {

       // validation check if emoji is not null
   ```

```
    if (detectedEmoji == null) {
      reset();
    }
    if (!isDetecting || startTime == null) return;

    var start = new Date();
    var timeElapsed = start.getTime() - startTime.getTime();

    if (timeElapsed < 10000) {
      if (detectedEmoji === targetEmoji) {
        score += 1;
        total += 1;
        scoring();
      }
    } else {
        total++;
        scoring();
    }
}
```

3. The code then updates the score and show the next emoji

```
function scoring() {
    setScore(score, total);
    targetEmoji = generateEmoji();
    setTargetEmoji(targetEmoji);
    startTime = new Date();
}
```