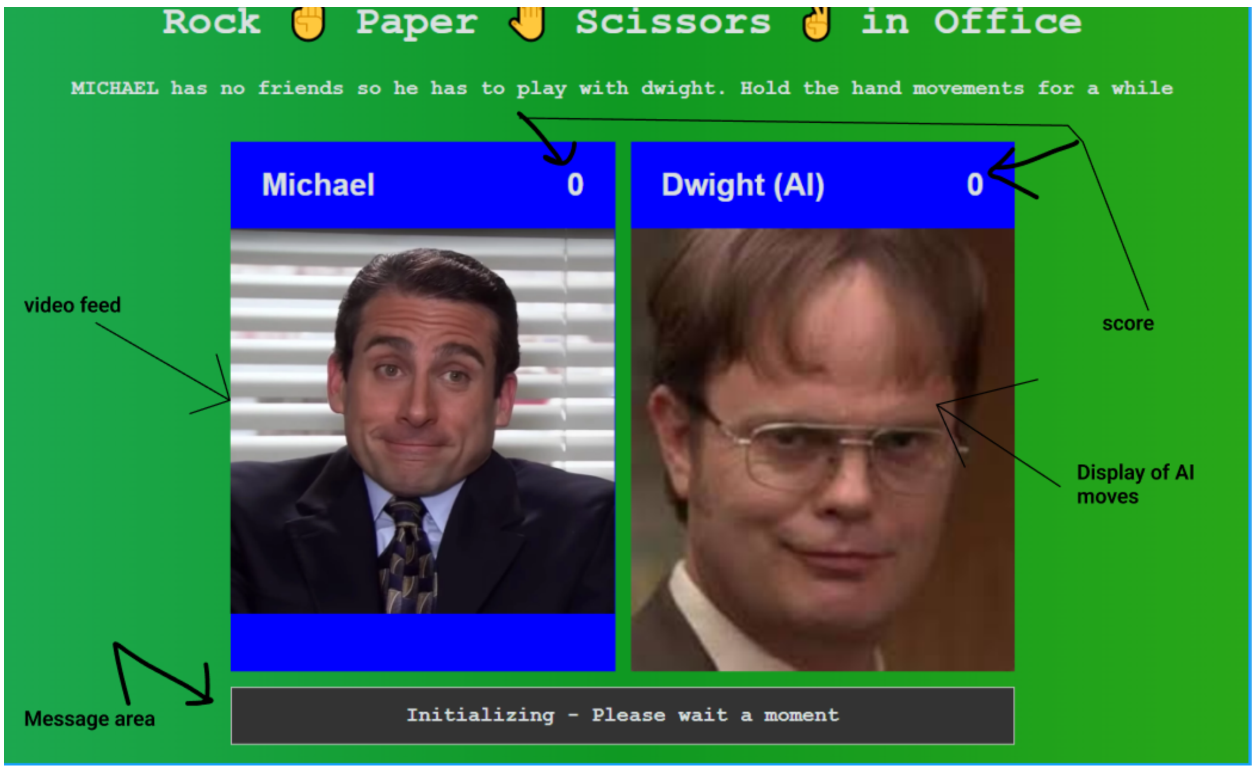
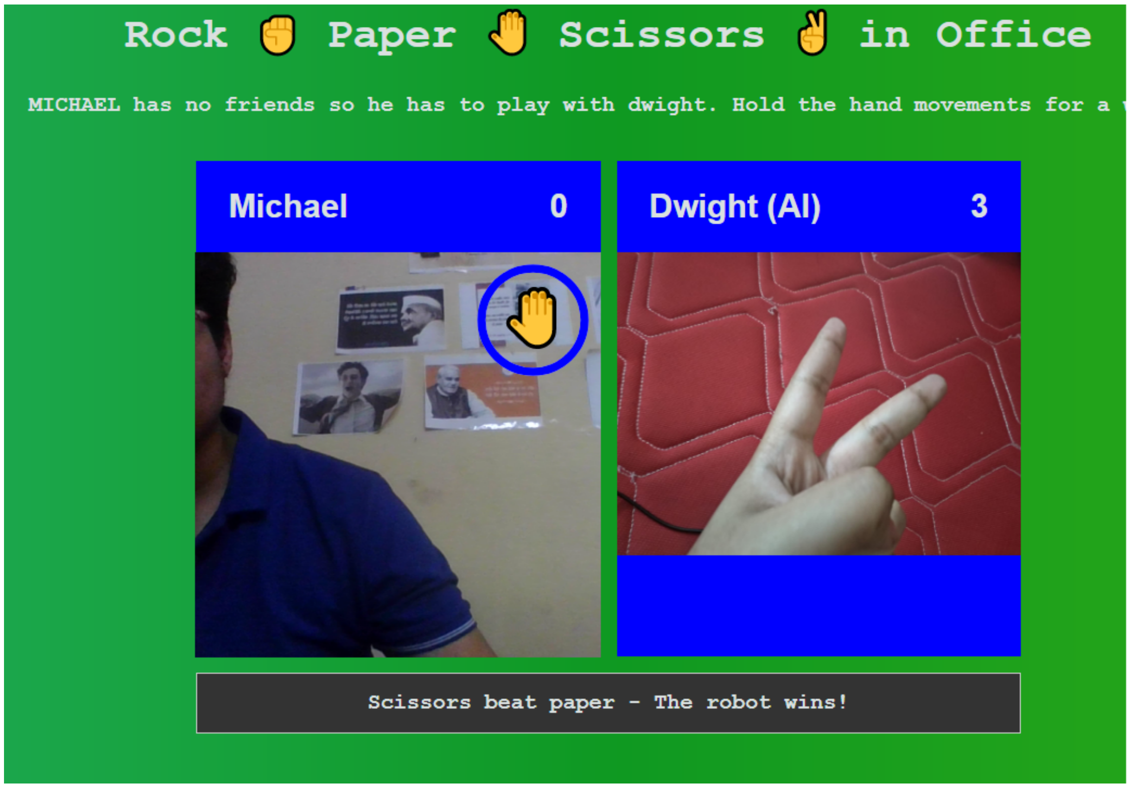
Name: Gyanendra Prakash
Roll : 190103110
College : IIT Guwahati

Overview :

This project leverages tensorflow's MediaPipe Handpose which is a lightweight ML pipeline consisting of two models: A palm detector and a hand-skeleton finger tracking model. Based on this model I have used another library which estimates the curl and direction of individual fingers (pinky,index, thumb, middle,ring) to determine hand gestures.

UI Design :


Rock ✊ Paper ✋ Scissors ✌ in Office
MICHAEL has no friends so he has to play with dwight. Hold the hand movements for a while
Michael
0
Dwight (AI)
0
video feed
score
Display of AI moves
Message area
Initializing - Please wait a moment

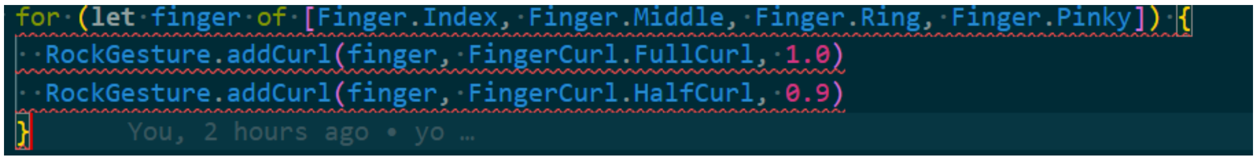

Rock ✊ Paper ✋ Scissors ✌ in Office

MICHAEL has no friends so he has to play with dwight. Hold the hand movements for a

Scissors beat paper - The robot wins!

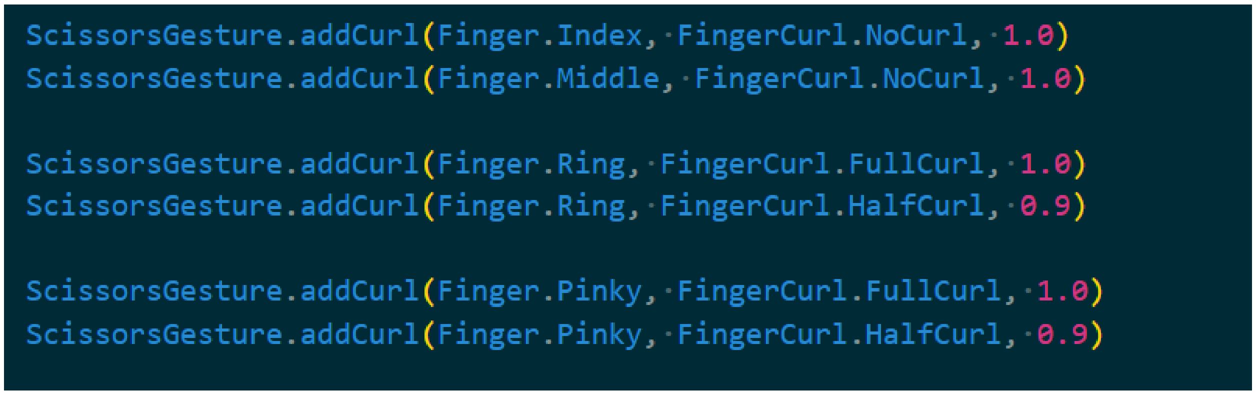
Logic behind gesture detection :

Describing rock gesture:


```
for (let finger of [Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky]) {
  RockGesture.addCurl(finger, FingerCurl.FullCurl, 1.0)
  RockGesture.addCurl(finger, FingerCurl.HalfCurl, 0.9)
}       You, 2 hours ago • yo …
```

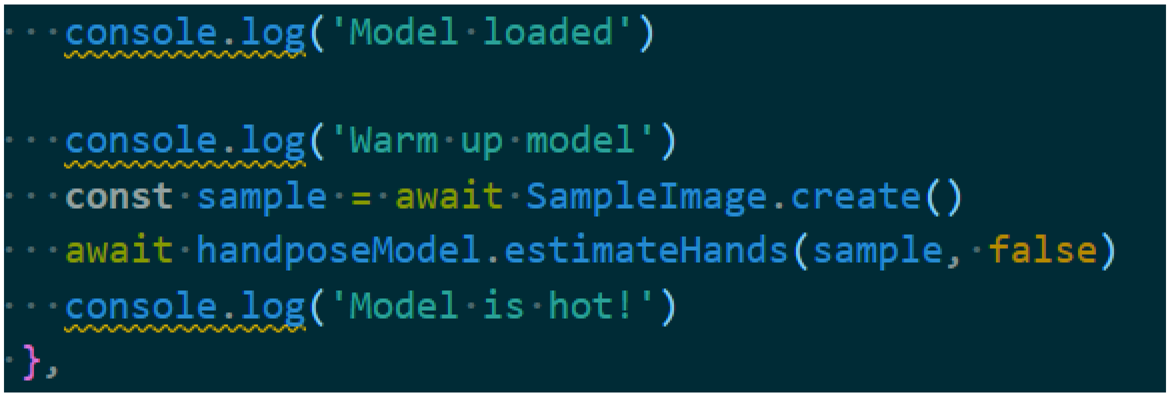
The rock gesture can be described as a condition when all fingers are fully curled. The 1.0 represents the match score, I have added a 0.9 score for half curl based on trial and error, the thumb particularly never fully curls in case of rock gesture.

Similarly scissors gesture has been described as follows:

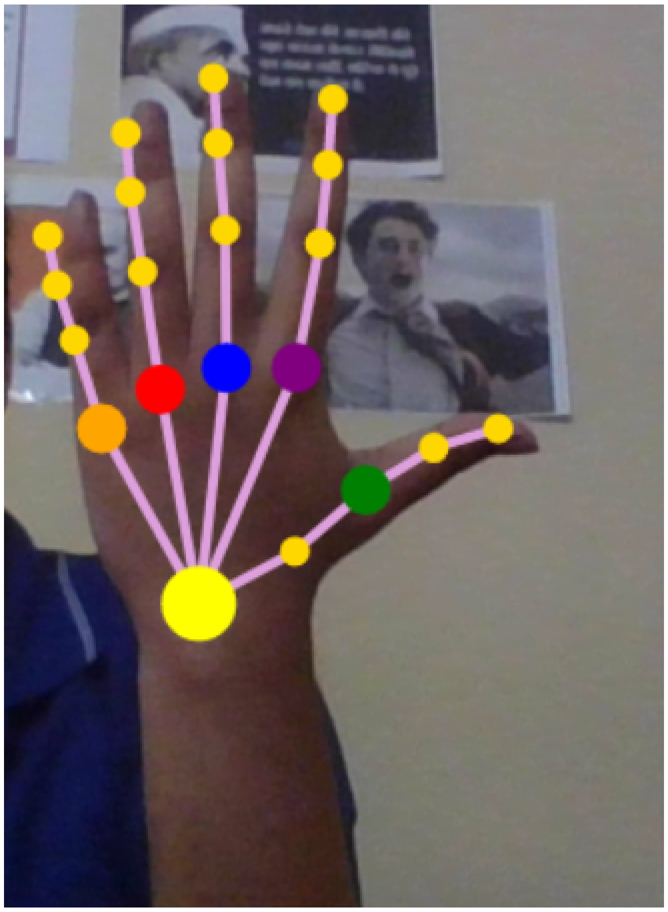


```
ScissorsGesture.addCurl(Finger.Index, FingerCurl.NoCurl, 1.0)
ScissorsGesture.addCurl(Finger.Middle, FingerCurl.NoCurl, 1.0)

ScissorsGesture.addCurl(Finger.Ring, FingerCurl.FullCurl, 1.0)
ScissorsGesture.addCurl(Finger.Ring, FingerCurl.HalfCurl, 0.9)

ScissorsGesture.addCurl(Finger.Pinky, FingerCurl.FullCurl, 1.0)
ScissorsGesture.addCurl(Finger.Pinky, FingerCurl.HalfCurl, 0.9)
```

A warming up part for model has also been added to improve the user experience. Model is provided with a sample image at the early initialisation , to warm up so that later predictions can be faster



```
  console.log('Model loaded')

  console.log('Warm up model')
  const sample = await SampleImage.create()
  await handposeModel.estimateHands(sample, false)
  console.log('Model is hot!')
},
```



Video frame is processed as shown in this figure, the hand is divided into landmarks, on the basis of 21 key points gestures are scored, the gesture with highest match score is selected. To increase stability, detections from several consecutive frames are combined to produce a single result.

Design of Scoring and message display

```
const randomNum = Math.floor(Math.random() * gestures.length)
```

Random number generator determines the move of AI, where gesture is the length of the array of possible moves. A function generates the random gesture through this.

Winning and losing over each other has been set by if-else block. A UI.js function based react component was made to which we can pass, parameters which determine which image will be shown on AI side depending upon the result returned by gesture generator function. The message is also fed to that where it is transferred to the html template using query selector.

Possible full stack generic approaches

Initially I had thought to approach the problem in several ways, I had thought of training a custom model, and uploading it after converting it into a model.json format and hosting it on a cloud service storage. That seemed tedious.

A full stack approach would have been to collect data through a node.js app, and store it in a database (Amazon s3 buckets/Gcp buckets would store image data and meta data would be in a postgresql database) , which could have been used to train a model. The model would have been converted to a tensorflow model.js, and then subsequently used.