PROJECT DOCUMENTATION

```
{developer: "Ebrahim_Badawi", class: "cart351", prof: "Sabine Rosenberg"};
```

concept

It all started from a simple idea of having a website which makes all connected users smile at the same time; then it would capture their smile, create some sort of artwork or collage of all people smiling, and captions the result with an uplifting phrase like “we are more beautiful when we smile”; Nonetheless it all changed.

Questioning the concept and how to achieve what I desired, of course, with the invaluable help of Sabine Rosenberg, I wondered “how computer would even know what makes people smile at the first place?” this question has led the project to a different path, and I started to look into machine learning and how to teach a machine whether something is funny or not. So, I decided to make a website which uses users’ reactions to different things, for the start something like a joke, as training data in order to train itself and learn how to make people laugh.

Therefore, what you would see as my final project for this wonderful class is a foundation of website which ask user to tell a joke, it takes that joke and sees whether it is funny or not by telling it to other users and observing their reaction via computer vision. Next, in the backend, the website would start training itself with the data collected from all active clients, whilst in the frontend, users would be able to interact with each other by watching an almost real-time collage of all active clients faces, in other words, they can see others’ reactions, or one can say their facial response, to the joke they said.

Building the Machine

To build this project, being a kind of newbie, I had to learn all sorts of things, from a simple JavaScript syntax, to machine learning. In order to explain my project more comprehensively, I divided it into three sections: client/server communication, Face detection, and Machine Learning.

Client/Server communication

I used **Node.js** as an asynchronous event-driven JavaScript runtime, particularly because Node.js is designed to build scalable network applications. Node.js also eliminates the waiting, and simply continues with the next request. On top of that, I used **Express** module which is a

minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. Express also act as a router which makes life much more enjoyable for coding. As a quick note, routing simply refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

Another Node module I used is called **express.static**, which is a built-in middleware function in Express to serve static files such as images, CSS files, and JavaScript files to clients.

Last but not least, I used **Socket.IO** library to complete my client/server communication section. This powerful library, Socket.IO, enables real-time, bidirectional and event-based communication between clients and servers, which makes everything more comprehensible.

Having my server and clients communicate in an event-based manner is actually a foundation upon which my whole system is built and made everything possible to happen in real-time and b be super responsive.

Face Detection

In order to give my machine the ability to observe clients and see their reaction to a joke, I used a pretty interesting library called **face-api.js**. Face-api is a javascript module, built on top of tensorflow.js (a library for machine learning in JavaScript) core, which implements several CNNs (Convolutional Neural Networks) to solve face detection, face recognition and face landmark detection, optimized for the web and for mobile devices.

even though learning face-api's API was a bit tricky and confusing at first, with the help of this library, I was able to detect faces and have access to an array of facial landmarks' coordinates. By having this data in hand, I was then able to extract different section of the detected face in order to make a hybrid face out of every online client's face. Face-api also detects a limited set of expressions, among which "happiness" was the one I needed to evaluate whether a joke has landed and got the laugh or not.

Machine Learning

Since learning how to teach a computer and creating a neural network of my own was outside of the scope of my brain's abilities in the limited time, I have found a JavaScript library which make it easy for you, and it is called **Brain.js**. Brain.js is a GPU accelerated library of Neural Networks written in JavaScript for Browsers and Node.js. It is simple, fast and easy to use. Since face-api is already built on top of tensoflow.js, at first I thought it is better to use the same tool to build my neural network, but then I found that it may need much more time for exploring it and learning how to make things work in a way to help me get to my objective.

Nonetheless, now, I somehow regret not using tensorflow instead of Brain.js, and that is because Brain.js is very powerful for training a machine with a different set of numbers or 0s and 1s, yet when it comes to training with a set of strings, it lacks functionality. A simple example would be the fact that you do not have the option to have a stream of training data when you use strings as you training data; it basically means that you cannot have your machine constantly training itself and remembering its previous trainings, instead it has to retrain itself everytime.

That was when I decided that it would be a good idea if I have a database to store my training data so that I can access it any time to train my machine. Thus, I also used **SQLite3** module, which is a software library that provides a relational database management system, to create a database and store the training data from each day in a separate table so I can retrieve it easily afterwards.

Finally, everything came together and the machine's structure was made.

The Machine

Since my project is more of a backend of website and there it not much to show in its UI, I have provided extremely neat and pleasing to read comments in my code so one could understand how it works. Feel free to look at the code here, and here. Yet, since we are here, I would explain how it functions generally.

For making a hybrid face of all connected clients in real-time, every second server assigns each part of the hybrid face to a random active client. Then, simultaneously, every half a second server starts the client/server conversation to get the hybrid face parts from the assigned clients in order to send it to all other clients to display it on their screen. The conversation would look like the following:

- **server:** 'areYouReady'?
- **client:** [if it has the result from face detection =>] 'readyToSendParts'
- **server:** [checks if client should send a part to server, if yes, which part =>] 'partRequest' [sends the name of the part it wants from this client]
- **client:** [extracts the requested part(s) and sends a dataURL to server =>] 'gotMouth'/'gotNose'/'gotLeftEye'/'gotRightEye'
- **server:** [gets the part(s) and sends the dataURL(s) to other clients to display =>] 'displayMouth'/'displayNose'/'displayLeftEye'/'displayRightEye'
- **client**: [gets the dataURL of part(s) of other clients for hybridFace from server and displays it (them))]
- **OVER**

And for machine learning part, whenever a user submits a joke, the conversation starts:

- **client:** [listens for click on submit button, when it is fired, gets data from the text box and sends it to server =>] 'textChat'
- **server:** [send the received data to all other clients =>] 'jokeFromServer'
- **client:** [waits for 3000ms for client to read the joke then starts observing user's facial response to the joke for 4000ms. it calculates the average happiness of the user (with the help of face-API) and sends this number with the associated joke back to server =>] 'facialResponse'
- **server:** [save the data into the database and stores it into trainingData variable, with which, then, trains itself]

Challenges

Challenges are part of coding, and actually it is what makes coding more interesting. I would say every aspect of this project had its own challenges. From the beginning, finding an appropriate library for both face detection and machine learning was a huge challenge; just researching and learning them by coding a test pages, was super time consuming, but that is inevitable which you cannot avoid and ignore.

Then, the next thing that challenged me a lot was Promises and awaits in JavaScript. JavaScript being an asynchronous programing language and all event based client/server communication happening asynchronously, mixed with Promises and awaits which make things more synchronous was just too confusing. The real challenge is optimizing everything in a way to have almost real-time response.

Learning Node.js was a bit of a challenge in the beginning, but after getting comfortable with it, I have found that it made facing and overcoming other challenges much easier. The great example of it would be creating a secure context and an https server, which is needed to get user media from devices other than server, with self signed certificate and a key, which Node made it super easy to do with 3 simple lines of code.

What is Next?

Everything. What the machine is now in this stage is just a foundation of a greater thing. First of all, I am going to work on UI more to give it more of a personality, because now it does not look like anything and suggests nothing in particular in the first glance. Then I would start by optimizing the learning part the machine by looking into more options that I have for Neural Network libraries for javascript. Then I would like to structure the learning part more to specify what to learn and what not to learn, and then after it starts learning, the things you can do with the data and the trained machine is just enormous. The last thing that I would love to do is to

make the visulas a bit more abstract and more like a computer generated art works, which was my intention since the beginning.

Since training the machine needs data, as soon as I make its backend solid enough so that it would be ready to launch, I would like to make it online using Heroku and test it and start training it. Then what I would like to do is to make is compatible with mobile devices as well and I think it would get more traffic that way.