Project Goals:

The goal of the project was to make a web program that would utilize facial recognition for account security. Among my goals upon undertaking this project were to better understand how servers are made and operated, and to utilize my understanding of the material taught in this sequence to create the program. An additional goal was to force myself to gain a better understanding of web programming in general as my only previous experience were a few projects in a graphics course, with this project I would be forced to learn the more complex aspects of web development and html coding. I also wanted to make a focus for the project on the communication between different web pages in order for this to work. While there are some issues in the program regarding the database, the code does what I initially set out to do which was make a login function that depended on matching the user's face with the account to be logged into's face.

Project Design:

Originally, the project was supposed to use node.js as its framework and use the openCV package to operate the facial recognition code. Due to early and persistent complications there were issues with installing the openCV package and I resulted in utilizing the Face++ Api that was shown to me by Dr.Chen. This process had its own set of bugs when it came to uploading a picture to the cloud based storage that is run through Face++ but these were remedied when I included an additional Api, the Imgur Api, in order to load images to a webpage that I could acquire a url location from and send it to the Face++ servers. When designing the program I aimed to make it do a two major functions: Registration of an account for a user, and logging in to any account for a user. In both processes, the web browser would first request access to the user's camera, once a username and password were selected the user could take a photo of themselves. What happened here was the user hit the button to snap the photo which would call a corresponding function, depending on if they were registering or logging in that varied very little until the end of the process, this function would take whatever the current image was in the video playback element on the html page, so you could actually take a decent picture since you could see yourself this entire time, and draw the current image onto a canvas element that's normally used to do some quick, basic art projects in javascript. The function then would utilize the canvas image for the remainder of the code by creating a post request for the Imgur API's upload function. This request of the imgur server would result in a responding JSON document from the Imgur server which would reply with success or failed. If the response was a success the program would parse the resulting JSON document for the link to the image that is now on Imgur, the request would set up a callback function where the program completely diverted in terms of what is done. In the case of Registration, the code creates an API object for Face++

using my personal credentials, makes a request for the face++ api to detect if any faces exist in the link the code sends to it, in this case it's the link from the Imgur response. Once this request is made the Face++ side makes its own JSON based response and returns it to the client. The response from Face++ holds a lot of information including general details about the image, whether or not it detected a face in said image, details about the face detected ranging from gender, race, and specific details pertaining to the facial structure of the face in question, and most importantly it responds with a face ID that the api creates to associate with the face it found for future reference. In the case of registration we want to save this id for future references to it. After the JSON response is received with a success of whether the request was processed or not, the program parses through the response and finds the face ID, if in this process it doesn't find one it actually tells the user that a face wasn't detected and refreshes the page for the user to try again after the user acknowledges the notification. Then, it passes the ID of the face and the username and password the user entered to another function. This function requests that the Face++ api creates a new person slotted under the storage of my application on there, that only I have access to, via another request to the api creating a person with the username the client inserted and the face ID extracted from the previous detection process. Once the person is made the code adds the username, password and face ID strings to its effective database, due to time concerns from working on other bugs from this code I was unable to get a more legitimate server side database running though I worked on researching it to get it to work, as a result this 'database' ended up being the browser's local storage using the username as the key and string the other two pieces of information as a stringified JSON document. This is one thing that I will most likely aim to fix after the semester ends for my own desire to make this project even better. Once the data is saved the program sends the user to the home page in order for them to attempt a login.

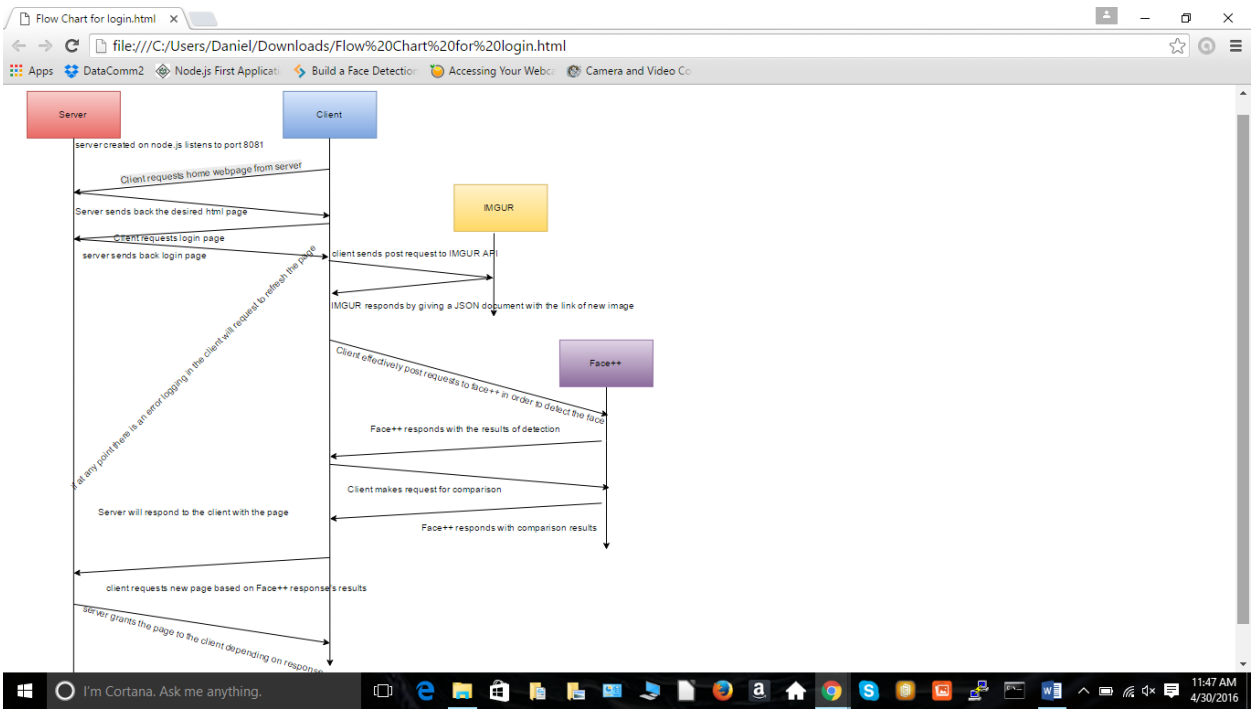

Figure 1: Here is a flow chart that depicts the server and clients requests and responses with each other and the two api's that operate on their own websites. If this is difficult to read, the html file where this is stored is in the repository named "Flow Chart for login.html".

On the login side of this operation we first check with the database we have stored in the local storage, if the username entered by the client is found as a key we check the key's corresponding password and compares that to what the user put in. If either of these fail the user is prompted about the error and the page is refreshed by the server for the user to try again. Once a match of username and password is made, here is where the image from the camera is taken and drawn onto the canvas. Like registration, the program sends an ajax post request to the imgur api, extracts the image link from the responding JSON, and sends it to a more detailed function for the face++ api's use, this time the callback functions are using, image link and the username as parameters. Next, the code makes the detection request from the api using the imgur link passed into the function. If a face is not detected, the client is prompted and the page refreshes. If a face is detected the code extracts the face id of the attempt image and the passes that and the username to one more function. After this has been done, the code now calls the recognition/compare function from the api and sends the request using the face ID of the attempted login image and face ID saved within that username's data that the code has been passing this entire time. The responding JSON document holds similarities between certain elements of the two faces along with a general float number of the two faces' entire similarity. The code extracts the general similarity of the faces and checks whether it is greater than or less than 70%. If it's greater than 70% we prompt the user that access was granted and send them to a page that would hold account information, if it is less than 70% we prompted the user that access was denied and refresh the page for an additional attempt. Above in figure 1 there is a diagram that details the order and requests of login process across the server and the APIs involved.

Some bits to note about the code. The server this program is run on is made via node.js. Due to the asynchronous nature of ajax requests and the face++ requests this program had created a large amount of nested callback functions which in my thoughts cause the program to be so slow at times but these are needed due to the reliance on information gathered from every request calls responding JSON document. Future updates to this would probably use mongoDB as the database source as it has a package that can run with node.js. There is a video demonstration in the repository called 'project_demo.avi' the video isn't the best quality but if you'd like to see how some of it works I included examples of a success, a failure, a failure to detect faces in the input image due to the lighting of the room, and a mismatched username/password response.