

**Facial Recognition**

Client: Chris Dore, Algonquin College

**Prepared By: Green Bit**

Maryum Awan

Nicolas Daigle

Felipe de Paula

Jacob Jewell

Ahmad Kaafi

Aman Multani

April 2019

# Table of Contents

Contents

[Table of Contents 2](#_Toc6518340)

[Table of Figures 4](#_Toc6518341)

[Acknowledgements 6](#_Toc6518342)

[Introduction 6](#_Toc6518343)

[Project Overview 6](#_Toc6518344)

[Team members 7](#_Toc6518345)

[Stakeholders 7](#_Toc6518346)

[Summary Research 7](#_Toc6518347)

[Potential Languages 7](#_Toc6518348)

[GUI Frameworks 7](#_Toc6518349)

[Libraries used 7](#_Toc6518350)

[How Does Face Recognition Work? 7](#_Toc6518351)

[What is it used for? 7](#_Toc6518352)

[What are the cons? 8](#_Toc6518353)

[What does a simple application look like? 8](#_Toc6518354)

[What does a complex application look like? 8](#_Toc6518355)

[Specs/General Info 8](#_Toc6518356)

[Implementation Steps 8](#_Toc6518357)

[Psychological View 8](#_Toc6518358)

[Verbal/Non-verbal disconnect 8](#_Toc6518359)

[Throat-clearing/Swallowing 9](#_Toc6518360)

[Face Activity 9](#_Toc6518361)

[Hand Activity 9](#_Toc6518362)

[Eye Studies 9](#_Toc6518363)

[Grooming Gestures 9](#_Toc6518364)

[Technology Used 9](#_Toc6518365)

[Requirements elicitation techniques 9](#_Toc6518366)

[Engaging 9](#_Toc6518367)

[Technical 10](#_Toc6518368)

[State Diagrams 10](#_Toc6518369)

[High Level State Diagram 10](#_Toc6518370)

[Low Level State Diagram 10](#_Toc6518371)

[User Stories 11](#_Toc6518372)

[User Interface Screenshots 16](#_Toc6518373)

[Final Product 18](#_Toc6518374)

[Acquiring a Data Set 20](#_Toc6518375)

[Version Control 20](#_Toc6518376)

[Meeting Schedule (Appendix) 21](#_Toc6518377)

[Sprint Overview 21](#_Toc6518378)

[Sprint 1 21](#_Toc6518379)

[Retrospective: 21](#_Toc6518380)

[Sprint 2: 22](#_Toc6518381)

[Retrospective: 22](#_Toc6518382)

[Summary 22](#_Toc6518383)

[Sprint Overview 23](#_Toc6518384)

[Sprint 1 23](#_Toc6518385)

[Retrospective: 23](#_Toc6518386)

[Sprint 2 23](#_Toc6518387)

[Retrospective: 23](#_Toc6518388)

[Sprint 3 23](#_Toc6518389)

[Retrospective: 23](#_Toc6518390)

[Summary of individual contributions 23](#_Toc6518391)

[Summary of what the team accomplished 25](#_Toc6518392)

[**Task:** Blink Detection 25](#_Toc6518393)

[**Task:** Add time counting the seconds when eyes closed 27](#_Toc6518394)

[**Task:** To implement nod detection and head movement 29](#_Toc6518395)

[**Task:** Detecting Mouth Movement 31](#_Toc6518396)

[**Task**: Recognize speech input from the microphone 37](#_Toc6518397)

[**Task:** Detecting and retrieving the audio during the video stream 37](#_Toc6518398)

[**Task:** Detecting and retrieving the audio during the video stream 39](#_Toc6518399)

[**Task:** Creating the splash screen 40](#_Toc6518400)

[**Task:** Determining the color in front of the Webcam 40](#_Toc6518401)

[**Task:** Ability to record the video stream with the full functionality of different function 42](#_Toc6518402)

[**Task:** Save Facial Amalyses Data 42](#_Toc6518403)

[**Task:** Design Application Log 44](#_Toc6518404)

[**Task:** Design 44](#_Toc6518405)

[**Task:** Design UI 45](#_Toc6518406)

[Lessons learned 46](#_Toc6518407)

[Test Plan 47](#_Toc6518408)

[We put a lot of focus into the red quadrant, as our main testing is done by simulations 47](#_Toc6518409)

[References 47](#_Toc6518410)

# Table of Figures

[Figure i: AI Facial Recongnition State Diagram 9](file:///C:\Users\jakej\OneDrive\Documents\Algonquin\2019W\FinalReport.docx#_Toc6518411)

[Figure ii: Low-level State Diagram 10](#_Toc6518412)

[Figure vi: Sprint 2 21](#_Toc6518413)

[Figure vii: Summary of all sprints 21](#_Toc6518414)

[Figure viii: facial landmark 24](#_Toc6518415)

[Figure ix: eye\_aspect\_ratio Function 25](file:///C:\Users\jakej\OneDrive\Documents\Algonquin\2019W\FinalReport.docx#_Toc6518416)

[Figure x: Using the eye coordinates 25](#_Toc6518417)

[Figure xi: Determining Blinks 26](#_Toc6518418)

[Figure xii: Eyes closed 26](#_Toc6518419)

[Figure xiii: Eyes opened 26](#_Toc6518420)

[Figure xiv: Importing the time 27](#_Toc6518421)

[Figure xv: initializing the counter 27](#_Toc6518422)

[Figure xvi:Starting the counter 27](#_Toc6518423)

[Figure xvii: Displaying the counter and eye condition 28](#_Toc6518424)

[Figure xviii: Eyes open 28](#_Toc6518425)

[Figure xix: Eyes closed 28](#_Toc6518426)

[Figure xx: Set the variables 29](#_Toc6518427)

[Figure xxi: head\_mouvement Function 29](#_Toc6518428)

[Figure xxii: Nodding right to left 30](#_Toc6518429)

[Figure xxiii: Nodding left to right 30](#_Toc6518430)

[Figure xxiv: Mouth closed 31](#_Toc6518431)

[Figure xxv: Mouth open 31](#_Toc6518432)

[Figure xxvi: Mouth closed with the line 33](#_Toc6518433)

[Figure xxvii: Mouth open with the line 33](#_Toc6518434)

[Figure xxviii: The mouth array 33](#_Toc6518435)

[Figure xxix: Eye landmarks 34](file:///C:\Users\jakej\OneDrive\Documents\Algonquin\2019W\FinalReport.docx#_Toc6518436)

[Figure xxx: Mouth landmarks 34](#_Toc6518437)

[Figure xxxi: Speech\_recongnittion code 36](#_Toc6518438)

[Figure xxxii: Speech recording 37](#_Toc6518439)

# Acknowledgements

**Client**:

* Chris Dore, Algonquin College

**Consulting Firm**:

* Nicolas Daigle (Team Leader)
* Maryum Awan
* Felipe de Paula
* Jacob Jewell
* Ahmad Kaafi
* Aman Multani

**Project Professor**:

* Todd Kelly, Algonquin College

**External Resources**

* Adrian Rosebrock, PyImageSearch, Deep Learning Tutorials <https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>
* Peter Cunha, Emotion Detection, MIT License git source.

# Introduction

## Project Overview

We have chosen this project to have the opportunity to learn a new language and get an introduction to machine learning. In the process we plan to get familiar with Python, OpenCV, Tensorflow and Dlib to produce an AI facial recognition system that can accurately detect a person and their facial features.

Gathering data from facial features using mathematical functions and deep learning algorithms. With more data and proof using machine learning, we could potentially achieve a prediction whether the person is lying or not

Project Scope

The reason why we are designing this project is to improve the lie detection forms in today’s technology. The idea is to use a functional and accurate AI facial recognition scanner that can predict if a suspect is telling the truth or lying based on what we could gather through video streaming. The AI will locate an individual in a video or in real time and track their facial movements (eyes, blinks, facial twerks, etc.) As well as using audio recognition during an interview to give the AI more to work with when making the prediction. Once that has been achieved, the AI will then compare data that was gathered from analysing the suspect against a dataset of normal human facial movements as well as abnormal facial movement. The analyzation of the AI will result in a predicted calculation that will help determine if an individual has lied during a specific question. To accomplish this, we have designed an application which has a seamless UI, accessible by any user. Controlling a multi-featured video processing system, giving users the option to gather data from facial features, predict their emotion and detecting colors. For our acceptance criteria, we are expected to deliver an application containing features to help “predict a lie”, and to package the application up with a UI for the user.

# Team members

The Green Bit AI Facial recognition team consists of the following members

* Nicolas Daigle (Team Leader)
* Maryum Awan
* Felipe de Paula
* Jacob Jewell
* Ahmad Kaafi
* Aman Multani

# Stakeholders

Chris Dore is a Professor for Business Management and Entrepreneurship at Algonquin College. In the past he has worked internationally gaining experience worldwide by working in South Korea as a Government official, Public Relations Manager at Kia Motors, and as an Investor Relations Manager for L.G. Philips LCD. Amongst many employment achievements, in 2010 he became the founder and CEO of BizZen, a company which assists business owners in the selling process of their business. With stops along the way in equity research, financial services as a wealth advisor, and consulting; Chris now spends his time doing what he loves most, learning about new technologies (AI and Blockchain), teaching and creating robust businesses that provide value. Today he is part of the team developing the College mentoring program for young entrepreneurs.

“I teach because I love teaching and it allows me the opportunity to make a difference in my students lives. What I love most about teaching is inspiring students to pursue their hopes and dreams. I do this by creating classroom moments that engage students while creating a positive and fun learning environment.” - Chris Dore

# Summary Research

Potential Languages - Python,OpenCV, MatLab, C/C++.

GUI Frameworks - Tkinter, PyQt

Libraries used - OpenCV, Dlib, Tensorflow

## How Does Face Recognition Work?

The video stream is broken down and compared pixel by pixel to attempt to detect a face in the video stream. Using OpenCV, Computer Vision to manipulate the image into a scalable image for use when processing.

Using Dlib libraries we can use mathematical functions and the landmark dataset to gather data from the face.

## What is it used for?

* It is mainly used by Law enforcement to help identify criminals
* Airports use it to also identify criminals and to compare the passport photo with the person to confirm their identity.
* Could be used for statistical purposes, ie. Tracking how many people are in a specific area each day

## What are the cons?

* Bad quality images or poor lighting would give less accurate results
* Showing emotions like smiling or a face a different angle would lead to less accurate results
* Covering parts of the face like clothing (scarves, hats etc.), having a different hairstyle and having makeup on would give less accurate results.

## What does a simple application look like?

Still image facial recognition with a generic identifier.

## What does a complex application look like?

* Real time facial recognition
* Search through databases scanning for a match
* Cut images and Save snippets
* Implement automatic scanning and if match found, sends alarm or trigger to external source.

## Specs/General Info

* LWIR (Long wave thermal infrared)
* Stats based on accuracy of different types of scanning –
* LWIR - 93.93%
* Visual Cameras - 97.05%
* Fusion - 98.40%

## Implementation Steps

* Algorithm to match a facial image
  + Find a face in an image
  + Analyze facial features
  + Compare against other faces
  + Make a predictable match - further increase accuracy
* Face Landmark Estimation
  + 68 facial pinpoints – landmarks
  + Bottom of chin
  + Outside edge of each eye
  + Inner edge of each eyebrow
  + Outline of Lips
  + Mid Nose and Span

## Psychological View

Cues to look for when someone is lying

### Verbal/Non-verbal disconnect

In terms of disconnect watch how a person nods their head in a affirmative motion while saying “No” or turns head side to side in a negative manner while answering “Yes” towards the question. Acting the opposite as what their words are portraying.

### Throat-clearing/Swallowing

Swallowing/Clearing their throat before answering the question could be a sense that they were just caught with a tough question and feel like a pit in their throat before they attempt to answer. Also, physiologically the question could spike anxiety which can cause dry mouth

### Face Activity

Often in the form of biting or licking lips

### Hand Activity

Rubbing or fidgeting with their hands

### Eye Studies

* Rapid Blinking
* Movement of the eyes back and forth
* Closing eyes during response which does not require reflection and time to answer

### Grooming Gestures

* Male – Fixing a tie, shirt, cuffs, glasses
* Female – Movement of putting hair behind ear

Some studies say sweat is not an issue, but the act of using a handkerchief or cloth to wipe the sweat away may be a sign.

Adjusting objects around themselves, “You ask a question, and suddenly the phone isn’t turned the right way, glass of water is too close, or the pencil isn’t in the right place”

# Technology Used

Our application has many different functions built in to change the processing state, we use Dlib libraries for our facial analysis data to generate mathematical calculations on the face to retrieve different data on certain facial features.

For Emotion detection, processing the video stream using Tensorflow and pre-trained algorithms to determine which emotion the user is displaying.

With our main technology being OpenCV to retrieve video stream from the webcam and manipulate it into a scalable image for processing.

All of this is wrapped in a GUI using Tkinter frameworks to give the user a visually appealing and easily accessible UI.

# Requirements elicitation techniques

The following is a set of questions which we asked Chris Dore on our first meeting

## Engaging

1. Vision?
2. What is the end objective?
3. How is the product going to be used?
4. What external resources will we need? Camera (Thermal)? Database?
5. What do you see as the biggest risks the team needs to consider?
6. How do you like us to communicate with you by email, weekly meetings or both?
7. Is there anything we should learn before we start the project?

## Technical

1. Will it be automatically scanning real time imaging?
2. Camera created images to then match faces, or no camera involved and just image facial matching.
3. Will the application be used to scan images or video or both?
4. What type of imaging, Thermal? Visual?

# State Diagrams

## High Level State Diagram

Below is a high-level state diagram showing the flow of our software.

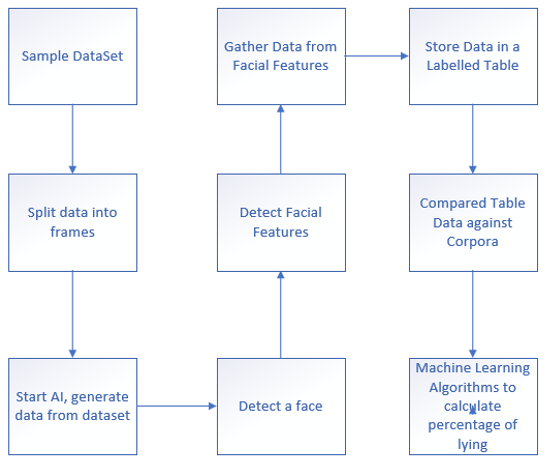


Figure : AI Facial Recognition State Diagram

## Low Level State Diagram

Below is a low-level state diagram showing the flow of data generation and retrieval

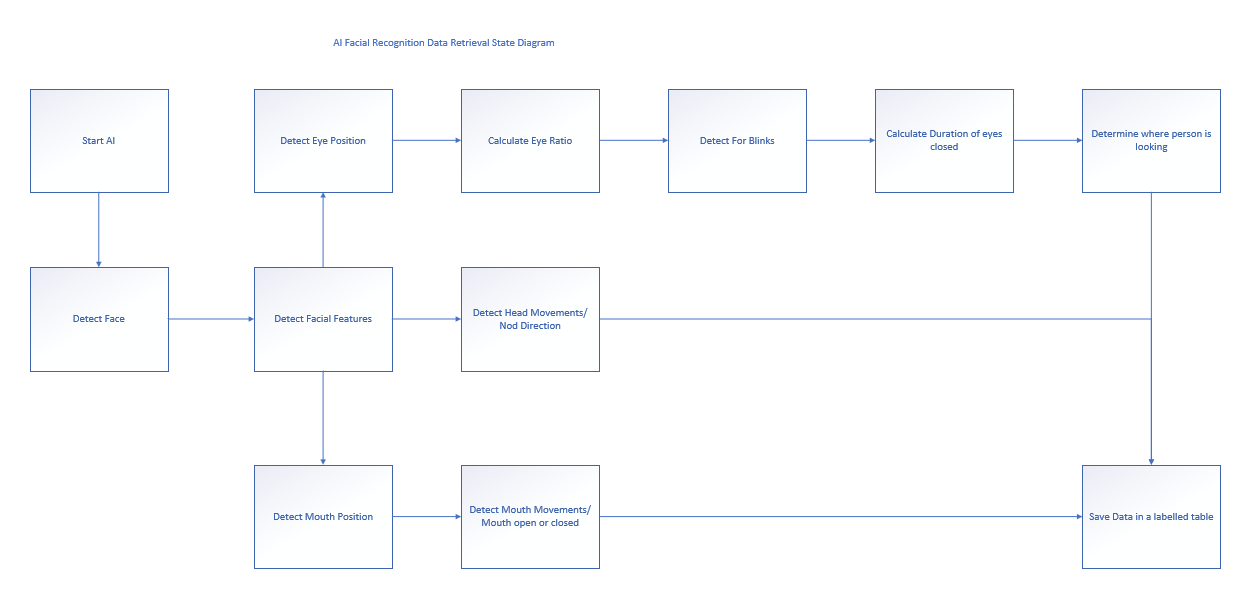


Figure : Low-level State Diagram

# User Stories

A set of user stories of which we have completed or are working on implementing in the future.

|  |
| --- |
| User Story:   * As a user, I want to be able to identify a face so that I can see if there is a person present in the frame. |
| Conversation:   * Having a program that can detect a face in different surroundings. * Able to compare the recognized face with faces contained in a database * Able to detect a face even when some of the face is covered (scarfs or hats) or at an angle. |
| Testing and Acceptance Criteria:   * Simulation based testing * Out of frame range testing |

|  |
| --- |
| User Story:   * As a user, I want to be able to pinpoint and differentiate facial features so that I can then inspect the person in more detail. |
| Conversation:   * Able to determine by pinpointing the eyeball alignment or cheek bone structure * Able to find each facial feature. * Able to extract facial structures by knowing the indexes of specific parts of the face (eyes, ears, nose, etc.) |
| Testing and Acceptance Criteria:   * Based on where the face is located at, we can generalize where the common facial features are * Simulation Tests * Head movement tests * Out of frame range tests |

|  |
| --- |
| User Story:   * As a user, I want to be able to identify the eyes of a face so that i can keep track of its movement and the count of blinks. |
| Conversation:   * Locating the eyes based at a generalized location of the face * Once a general location has been found, pinpoint the pupil (center) of the eye * Measure movement within the range of the eyeball. * Tracking where the pupil is looking * Determine the white of the eye and if the white goes out it is a blink * Able to determine how many time eye blinks in one-minute compare to a normal rate * Plotting the eye aspect ratio over time |
| Testing and Acceptance Criteria:   * Functional Tests * Simulation Tests * Lighting Test * Head Movement Tests * Face location range tests |

|  |
| --- |
| User Story:   * As a user, I want to detect which direction the user is nodding so that the AI can use this data to improve lie detection algorithm |
| Conversation:   * Locate the middle point of the nose * Calculate the arc thresholds on the video frame * Compare the middle point of the nose against the arc thresholds to determine which direction the user is nodding |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

|  |
| --- |
| User Story:   * As a user, I want to detect movement around the users mouth and whether the mouth is open or closed so that I can generate data and apply it to the AI lie detection algorithm |
| Conversation:   * Locate the mouth nodes * Draw a line connecting the nodes * Use the same calculation for detecting eye blinks * Detect when a mouth is opened |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

|  |
| --- |
| User Story:   * As a user, I want to detect and retrieve audio generated from the video stream that I can use this audio for further improvements on the lie detection algorithm |
| Conversation:   * Initialize and terminate PortAudio. * Queries and inspect the available PortAudio audio devices. * To record and play the audio, open a stream on the desired audio parameters. * Read audio data from the stream using pyaudio API’s. |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

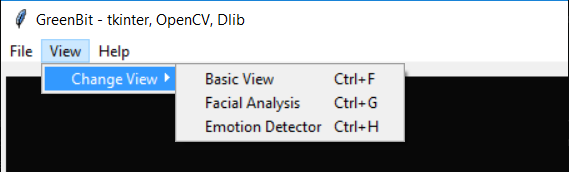
|  |
| --- |
| User Story:   * As a user, I want to detect which direction the user is looking so that I can retrieve this data for further use when applying the lie detection algorithm |
| Conversation:   * Locate the eye * Draw a line for the X axes for the eye * Draw a line for the Y axes for the eye * Determine where the X and Y axes are located relative to the eye outline * Detect where the user is looking |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

|  |
| --- |
| User Story:   * As a user, I want to count how long the user is looking in one direction during a question response so that I can use this data when applying the lie detection algorithm |
| Conversation:   * Apply the logarithm for detecting the eye ratio * Same as the eye blinks * And dividing into frames for sec * And counting the seconds |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

|  |
| --- |
| User Story:   * As a user, I want to count how long the user has their eyes closed during a question response so that I can use this data when applying the lie detection algorithm |
| Conversation:   * Import timer * If the eyes ratios are close to zero set timer to start counting the seconds * If the eyes are closed set message confirming the condition * Display visual message stating the value in seconds and eyes aspect |
| Testing and Acceptance Criteria:   * Simulation Tests * Unit Tests |

|  |
| --- |
| User Story:   * As a user, I want to store/save the gathered information so that the AI can have data to read and learn from. |
| Conversation:   * Encryption * External Storage to keep safe and secure * Having the capabilities to store the information that gets gathered after the analysis will allow our AI to have data to learn the patterns. * To store the data in to small pieces which will be easy to access |
| Testing and Acceptance Criteria:   * Security Tests * Performance and Load Tests |

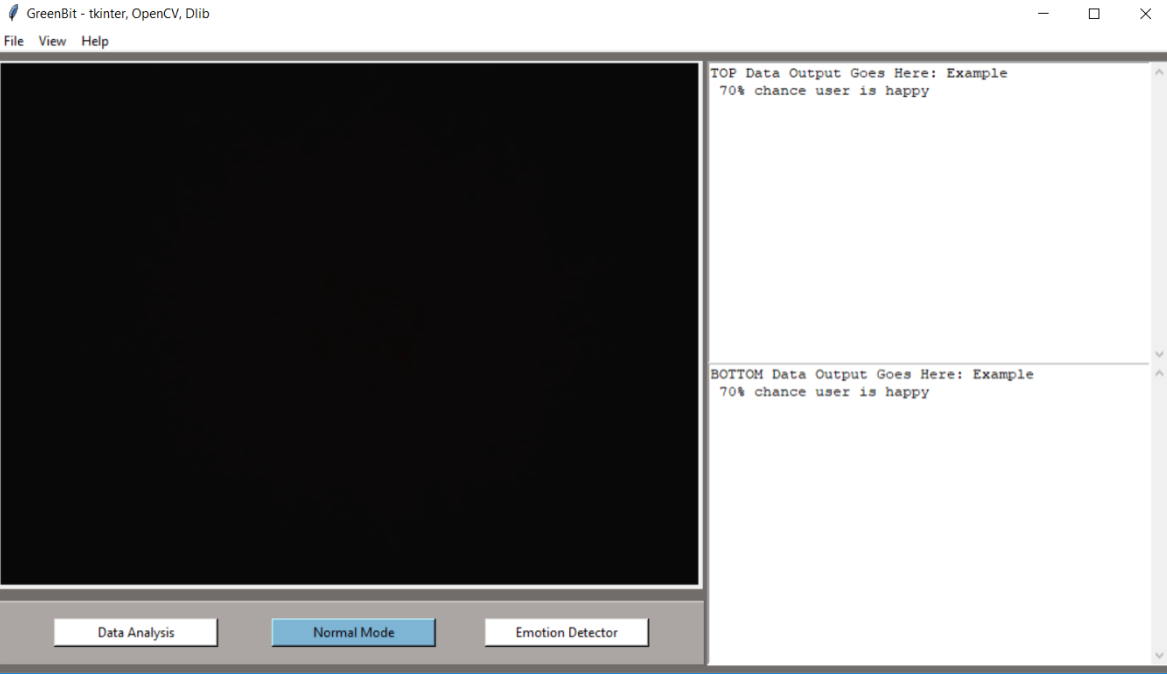
# User Interface Screenshots



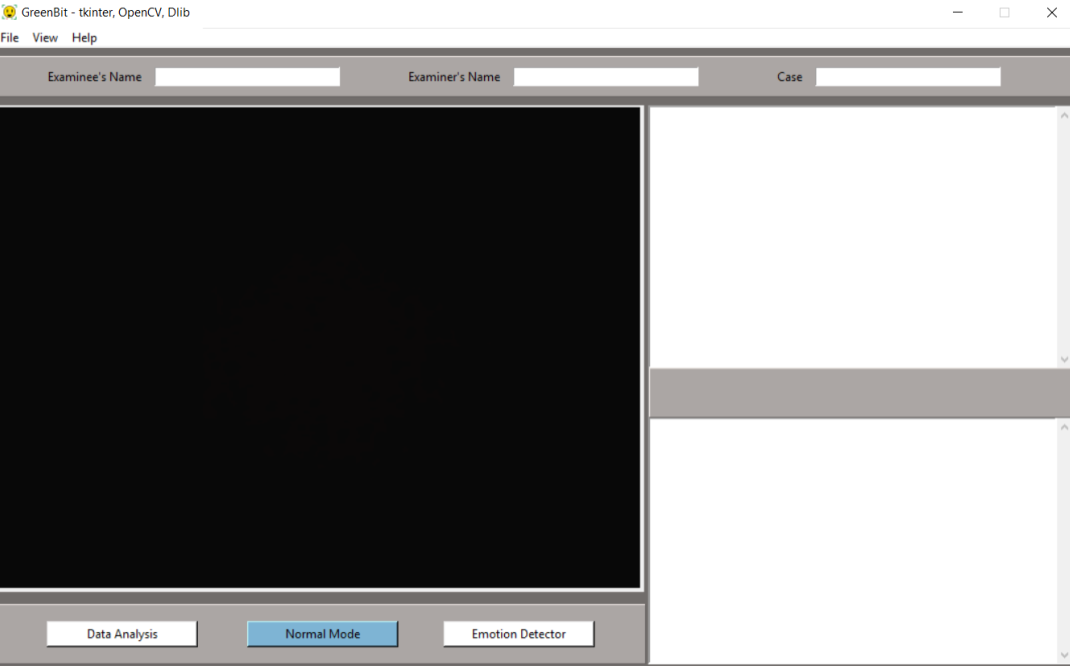
One of the initial things after getting a basic GUI working with the live camera feed was a simple menu bar, the first thing that we had added to it was the ability to change the camera views.



We then created a simple text box that covered most of the right side, do display information that is being generated from the analysis. We also placed 3 different buttons along the entire width of the bottom which allow the user to easily change what gets added to the video feed, meaning if it prints the emotion of the facial landmarks.



After reflecting on the GUI, we realised that it was best to have two different boxes to display the gathered information, since we have two different sources. The top box displays the data that gets generated from the emotion prediction.

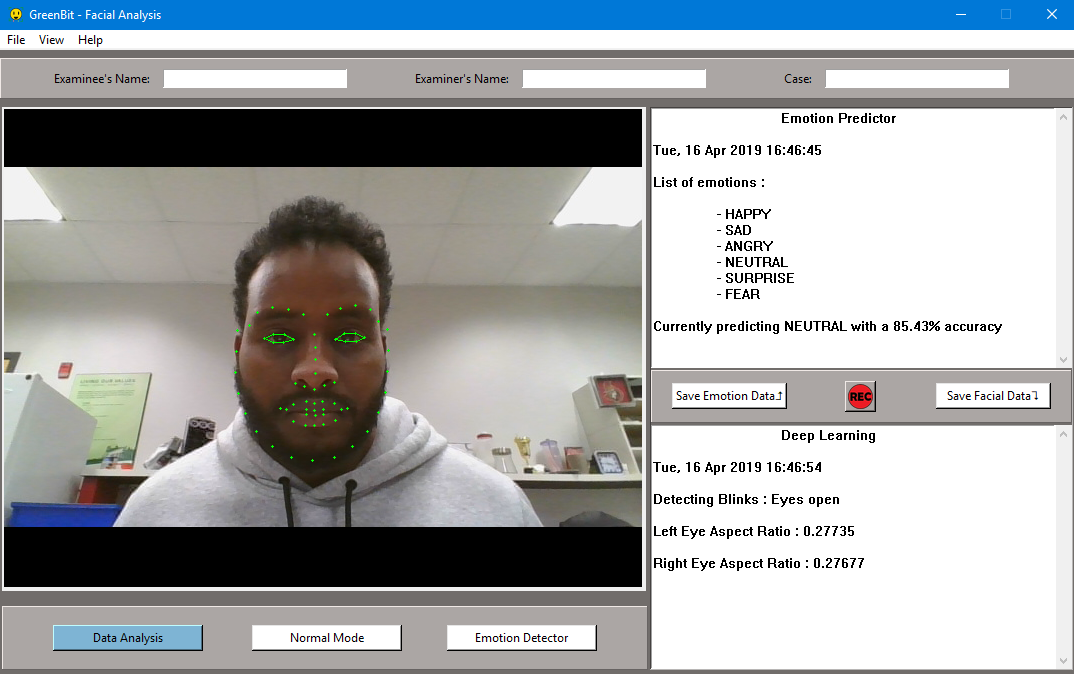


The next step was to first, add space between the two boxes so that we could place buttons for saving the data and for recording. We also added a top panel for the use of storing information, such as name and case number.

## Final Product



Once the Application launcher has been triggered, the user is greeted with a splash screen of the group logo and a loading bar. Once the splash screen has terminated, the main application will open.



The final version of the application can be visually divided into three parts, the top area, the view area, and the dialog boxes.

The menu bar which is located at the top of the application contains three main drop downs. Starting with the file dropdown, which gives the user the ability to open files, save the data, record a video and more. To the right of that, we have the view drop down, which is another way of changing what gets printed on the live video feed. And finally, we have the help drop down, which holds a pop up that describes the application. Just under the menu, we have implemented 3 different text boxes which are used to enter data of the event (name of the suspect, examiner and case number).

On the left-hand side of the application window, we display the live video input from the webcam in large for the user to see, under that canvas, there are 3 buttons that each trigger a different view mode which will print either the facial landmarks of the subjects face or print a square box around the subjects face with the predicted emotion.

On the right-hand side is where the data gets displayed. The top box is used to display the data that gets generated from the emotion predictor. Under the top box, we have three different buttons, one saves the emotion data, another one is to save the facial data and finally, we also have a record button. Under those buttons, we can see the second box with is used to display the live data that is getting generated from the Dlib facial landmark.

# Acquiring a Data Set

As the initial plan was to obtain a dataset which we can use to test our lie detector, we came up empty with our search for a dataset. Our next step was to create our own dataset by making videos of each other lying. How we have done that is through asking yes or no questions, that were unknown to the person being filmed with the purpose of surprising them. The person being filmed should or would occasionally lie on some of the questions and explain after which one they lied. This was a great starting point for us since we didn't have much experience in this type of field of acquiring data and it got our hands dirty, for a true solution to this dataset we would need huge corpora of individuals spanning to all races and sexes, to get a true dataset to base the lie detection system of off.

# Version Control

We are using git as our version control hosted on GitHub. We have chosen this system because it allows the team to easily share code and keep track of all the changes that every team member uses.

At the end of the team member’s task, the member uploads their code to our repository will the appropriate version number (i.e. Version 0.6), after the code has been pushed, the member will then update the README.md to include the new features that have been added to the program.

Code and Feature Testing

Most of our tests require us to use simulations to see if they work for example the aspect ratio of the eyes or the mouth are values that are changed infrequently (like blinks or opening the mouth), to capture that change we need to use a threshold. The threshold value is used to check for those infrequent changes but the mouth and the eyes vary from person to person so the threshold value must be effective in capturing those movements across everyone. We first had to see the aspect ratio of no movement and when there is. Once we have determined those boundaries, we can change our threshold based on when there is no movement. At our first test findings we found that either the threshold was low, and we were getting movement detected when we were not moving, or the threshold value was high, and we were not getting movement when we were moving. Once we got the threshold value to be right for one person, it did not work for the next person. We refined the value until we were confident that it can capture movements accurately.

# Meeting Schedule (Appendix)

We decided to meet two days a week outside of classes. Tuesdays we have a meeting with Todd Kelly where we talk about the progress of the projects and help plan and divide up the work into weekly sprints. We had a group meeting right before meeting with Todd so that we can talk about any problems we are having and how we will approach them. Monday meetings are about talking with the group about how the progress is going if a team member needs help or would like to make suggestions.

## Sprint Overview

## Sprint 1

### Retrospective:

During this sprint we have spent most of our time researching. Our first task was to find a dataset, but we were not able to find a dataset we can use. So, we decided on creating a video of each other, answering simple yes and no questions and trying to lie on some of them. After we have completed our videos, we then used it in our program, and we have successfully detected a face and when someone lied on any of the questions, we can add the facial data to a table.

## Sprint 2:

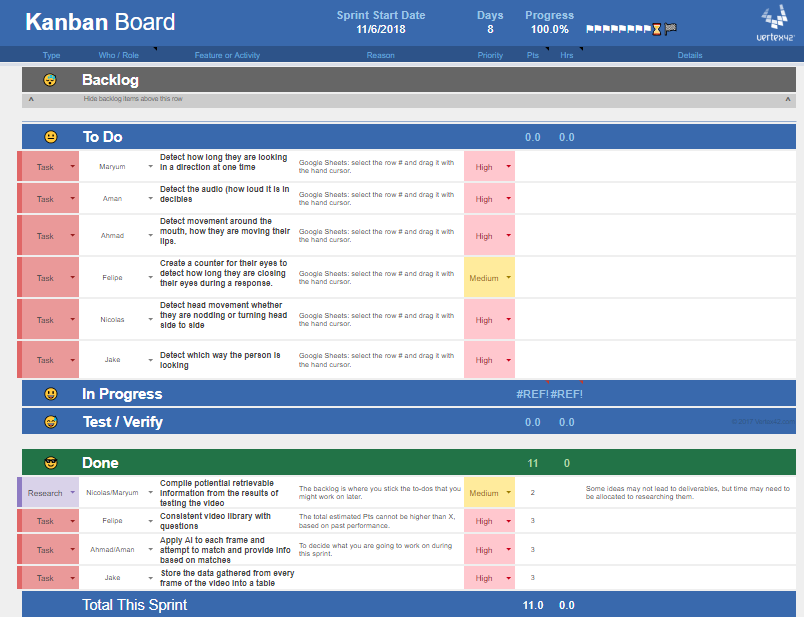


Figure : Sprint 2

### Retrospective:

In this sprint we focused on the main

## Summary

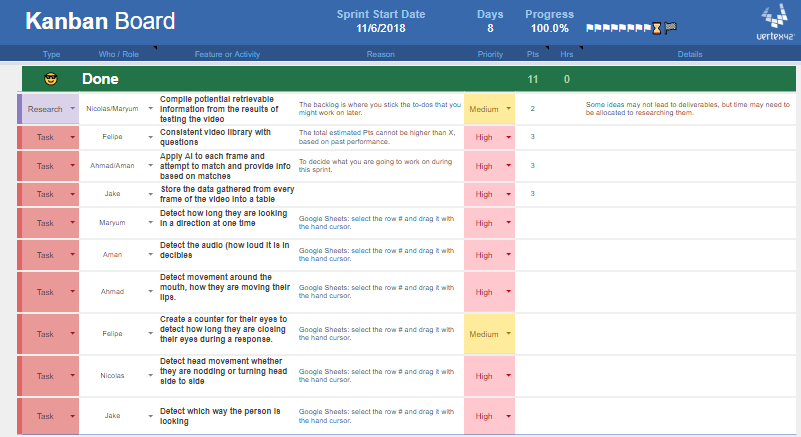


Figure : Summary of all sprints

## Sprint Overview

## Sprint 1

### Retrospective:

During this sprint, we spent most of the time researching how we are going to package and deploy our application, whether we would put the processing on the cloud with a local client, Or create an executable which the user can run locally resulting in the processing power being done locally.

With our scope we decided to focus on improvements and features and keep the application local in an executable.

## Sprint 2

### Retrospective:

In this sprint, we put emphasis on designing the GUI and completed the first stage prototype for our overall design. As well as features for the UI and overall application consisting of saving data generated from the emotion and facial data to a CSV file. We implemented a splash screen, which appears when the application is loading giving time for the back end of the application and to give the user visuals and confirmation that the application is loading.

## Sprint 3

### Retrospective:

In this sprint, we focused on packaging up the entire project and creating an MVP, Minimal Viable Product for our presentation and React showcase. With a focus on getting all features and documentation ready for handoff. With the final GUI design for our application, and features added and improved upon such as recording live video stream, adding metadata to the UI and generating a saved CSV file. Detection of colors added as a processing feature to show proof of concept and a start towards determining color on the faces in the live video stream. Which can lead to determining skin color changes such as blushing, stress changes, or fluctuation in internal temperature.

# Summary of individual contributions

An overview of individual contributions and responsibilities

**Nicolas Daigle**

* Early Research on how to detect a lie, and machine learning
* Setup AWS Server for testing and research whether we would keep the application locally or on the cloud. For our scope, we decided to stay local
* Discovered and Integrated an open source MIT license Emotion Detection trained algorithm
* Designed the GUI using Tkinter framework
  + Menu Bar
  + Scene/Layout
  + Top frame for metadata user input
  + Left side, Video Stream output
  + Right side, Text boxes for data from the video processing results.
  + Bottom frame for view control buttons

**Jacob Jewell**

* To store the data gathered from every frame of the video into a labelled table
* Detect the direction in which the user is looking
* Add code documentation in the project's source code
* Update the GitHub readme file to have working instructions to run the code (as a developer)

**Ahmad Kaafi**

* Apply AI to each frame and attempt to match and provide a prediction based on results
* Detect movement of the mouth and whether the user is opening or closing their mouth
* Obtain results from both Dlib capture and Emotion capture
* Display the analyzed data on the side of the webcam window in real-time
* Adding metadata to the results screen
* Adding the blink detection from our previous work to our application

**Felipe De Paula**

* To create some video corpora of simple yes or no questions to use alongside the AI when attempting to produce a lie prediction
* To count the length of time that the user is closing their eyes for, during a question response
* Save user examiner, examinee, and case to CSV file
* Save data buttons mapped to functions
* Save output results to a CSV file which can be opened and viewed on excel
* Implement a shortcut to save emotion data
* Design application icon
* Design record image for GUI
* Select best video stream format

**Maryum Awan**

* Early Research on how to detect a lie, and machine learning
* To count the length of time the user is looking in one direction
* Worked on displaying the emotions on the Web Application
* Storing the data from Web Application to the database
* Color detection
* Create a splash screen

**Aman Multani**

* Apply AI to each frame and attempt to match and provide a prediction based on results
* Detecting and retrieving the audio during the video stream for further use
* Record the video stream with the full functionality of different function
* Record the video stream with the full functionality of different function
* Determining the color in front of the camera which in future can be used for detecting the racial color and much more

# Summary of what the team accomplished

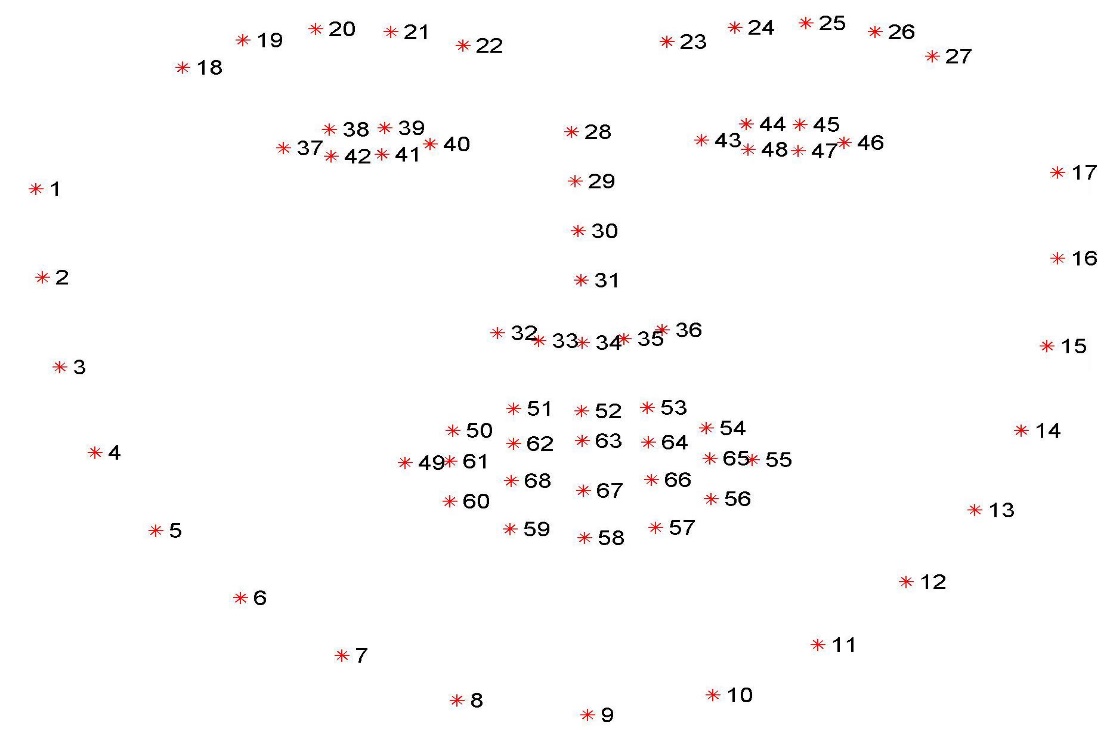


Figure : facial landmark

The above image outlines the facial landmark indexing api that we are using throughout our implementations. This landmark system allows us to easily extract coordinates for further use and to draw the facial feature on the video stream.

## **Task:** Blink Detection

**Author:** Jake Jewell

**Solution Outline:**

Allow the program to detect whether the eyes are opened or closed and keep track of the eye aspect ratio for analyzing purposes.

**Solution Detail:**

We define the eye\_aspect\_ratio function. This function takes in the (x, y) coordinates of the facial landmark for the eye.

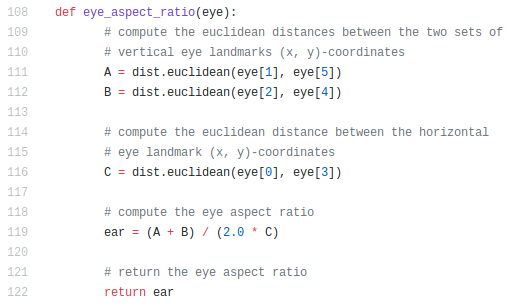


Figure : eye\_aspect\_ratio Function

The lines 111 & 112 calculate the vertical distance of the eye landmarks whereas line 116 calculates the horizontal distance of the eye. Finally, we then compute and return the aspect ratio of the eye in lines 119 & 122.

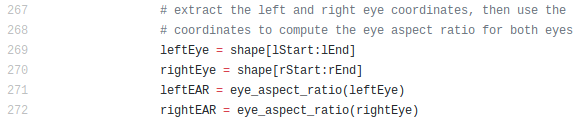


Figure : Using the eye coordinates

With that function, we are then able to get the ration of each of the eye by calling the function and assigning it to a variable.

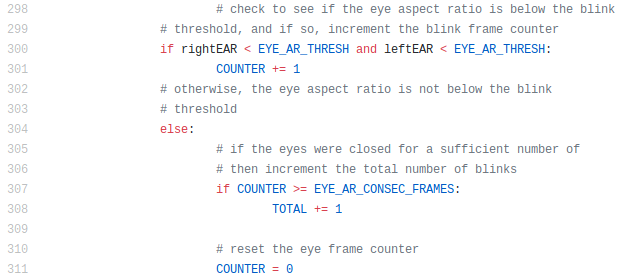


Figure : Determining Blinks

Once the aspect ratio for the eyes of the frame have been collected, we then check if the ratio is low enough to consider. The first if statement checks to see if the eyes have been opened, if so, then increase the blink counter by 1, otherwise, the eyes are closed, and we increment the total variable, with allows us to check if the eyes are closed for a minimum number of frames in order to consider that a blink has occurred.

|  |  |
| --- | --- |
| Eyes closed | Eyes Opened |
| Figure : Eyes closed | Figure : Eyes opened |

## **Task:** Add time counting the seconds when eyes closed

**Author:** Felipe de Paula

**Solution Outline:**

Detect if the eyes are closed by implementing counter and display message confirming eyes state (open or closed)

**Solution Detail:**

1. The first step was to import the actual timer on top of the page.



Figure : Importing the time

1. The second step was to add and initialize a START and END for the timer making the total value of seconds equal to the subtraction of both.



Figure : initializing the counter

1. A string variable called EYES was also added with the purpose of given the condition of the eyes.

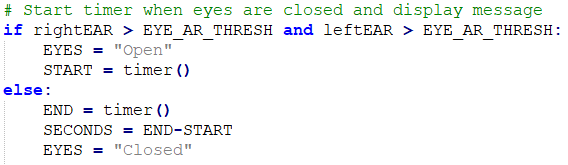


Figure :Starting the counter

1. Based on the aspect facial ratio values previously calculated, a condition was made for when the right and left eyes were open which would start the timer when eyes opened and end when closed. Also, the variable EYES was initialized to the string value open or closed depending on the condition of the eye.

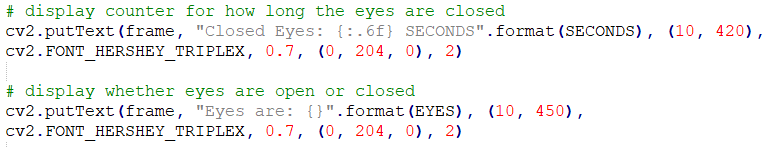


Figure : Displaying the counter and eye condition

1. To display the counter values at the screen the variable SECONDS (END-START) was added to the format.
   1. A second line of code was also implemented to display whether are eyes are closed or not.

**Display after Implementation**

|  |  |
| --- | --- |
| Eyes Open | Eyes Closed |
| Figure : Eyes open | Figure : Eyes closed |

## **Task:** To implement nod detection and head movement

**Author:** Nicolas Daigle

**Solution Outline:**

1. Split the frame into arcs, calculated based on the frame size
2. Find the x axis value of the middle point on the nose
3. Compare the continuously updating nose x axis value against the x axis values previous state and current arc thresholds
4. If change occurs save the current state and display a message to the user on screen.

**Solution Detail:**

1. The first steps to solving this problem is to ensure we have a dynamic calculations dependent on the frame size, for that we set a constant variable FRAME\_WIDTH so that if the frame size changes we only have to change it in one location and the rest of the program will follow its lead.
2. Next, we set the index we want to retrieve from the facial landmarks, in this case we chose index of 30 which will give us the middle of the nose.
3. Extract the coordinates of the nose using the function shape(int) which will return us the x and y coordinates of the index on the frame
4. After all steps are complete, we can then call our function.

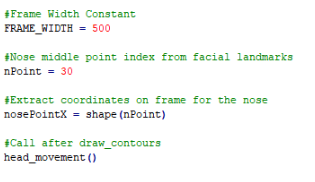


Figure : Set the variables

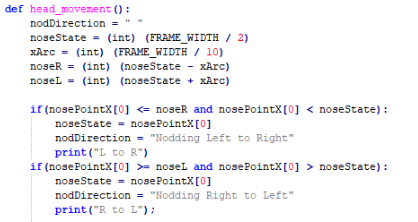


Figure : head\_movement Function

1. First step in the function is to calculate the arc threshold, nose middle point and left and right nearest threshold.
2. Compare the current nose x axis value against the previous state and the left and right thresholds.

|  |  |
| --- | --- |
| Nodding Right to Left | Nodding Left to Right |
| Figure : Nodding right to left | Figure : Nodding left to right |

## **Task:** Detecting Mouth Movement

**Author:** Ahmad Kaafi

**Solution Outline:**

1. Place a line on the mouth to indicate that it has been detected
2. Detect if the mouth has been opened in the frame

**Solution Detail:**

1. With knowing that we have already implemented the nodes on the face based on the facial landmark (in the Appendix) we first popped open the camera to test to see what happens when my mouth opens, and which specific nodes move when we open them.
2. We looked at the facial landmarks as seen above and noticed that there are two mouth sections, the outer mouth and the inner mouth. Based on what we saw in my testing of opening and closing my mouth we noticed that the inner mouth opens much more than the outer mouth.

|  |  |
| --- | --- |
| Mouth Closed | Mouth Open |
| Figure : Mouth closed | Figure : Mouth open |

By using the facial landmark index that has already been defined:

The mouth can be accessed through points [48, 68] or [“mouth”].  
 The right eyebrow through points [17, 22] or [“right\_eyebrow”].  
 The left eyebrow through points [22, 27] or [“left\_eyebrow”].  
 The right eye using [36, 42] or [“right\_eye”].  
 The left eye with [42, 48] or [“left\_eye”].  
 The nose using [27, 35] or [“nose”].  
 And the jaw via [0, 17] or or [“jaw”].

So, we chose to pick the nodes for the inner mouth which were (60-68, the inner mouth).

We used the following code to get the mouth index:

(mStart, mEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["mouth"]

**mStart**: being the start or the mouth array

**mEnd**: being the end of the mouth array

**face\_utils.FACIAL\_LANDMARKS\_IDXS**: contains the whole facial landmark index

**["mouth"]**: containing the nodes for the mouth

1. Now that we got all of the foundation set, we wanted to make sure the nodes were correct. To see if the mouth nodes we have picked were right we placed a line (similar to the eye we have) to the nodes we have picked.

We used the following code:

* Line 1: mouthFind = shape[mStartPlus:mEnd]
* Line 2: mouthMAR = mouth\_aspect\_ratio(mouthFind)
* Line 3: mouthHull = cv2.convexHull(mouthFind)
* Line 4: cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)

Notice the mStartPlus on Line 1? Since there wasn't a indexes for the inner mouth made we decided to make a new variable mStartPlus and made it equal to mStart plus 12 to get the value 60 and since mEnd was 68 which was the end of the mouth we did not need to change the value of that variable.

**mouthFind:** setwith the extracted coordinates of the mouth from the face in the frame

**mouthMar:** obtains the aspect ratio of the mouth based on mouthFind

We will talk about the function on Line 2 later.

**mouthHull:** set with the convex hull of the coordinates found by mouthFind

The convex hull is used to get the boundaries based on X coordinates, which is helpful since we need to see if the nodes we have picked were correct.

On line 4 we draw the line based on the current frame, the boundary (mouthHull) and the color which we used green (0, 255, 0).

After that was completed we then opened the camera to see if the inner mouth was detected.

|  |  |
| --- | --- |
| Mouth closed | Mouth Opened |
| Figure : Mouth closed with the line | Figure : Mouth open with the line |

Yes we have got the the line on the inner mouth. Our next step is to detect a mouth opening and count how many times the mouth opens.

1. On Line 2 we have a function call of mouth\_aspect\_ratio with a parameter of mouthFind which we used in the previous line. The array mouth entering this function contains the nodes of the of the inner mouth. The array holds a total of 8 elements.

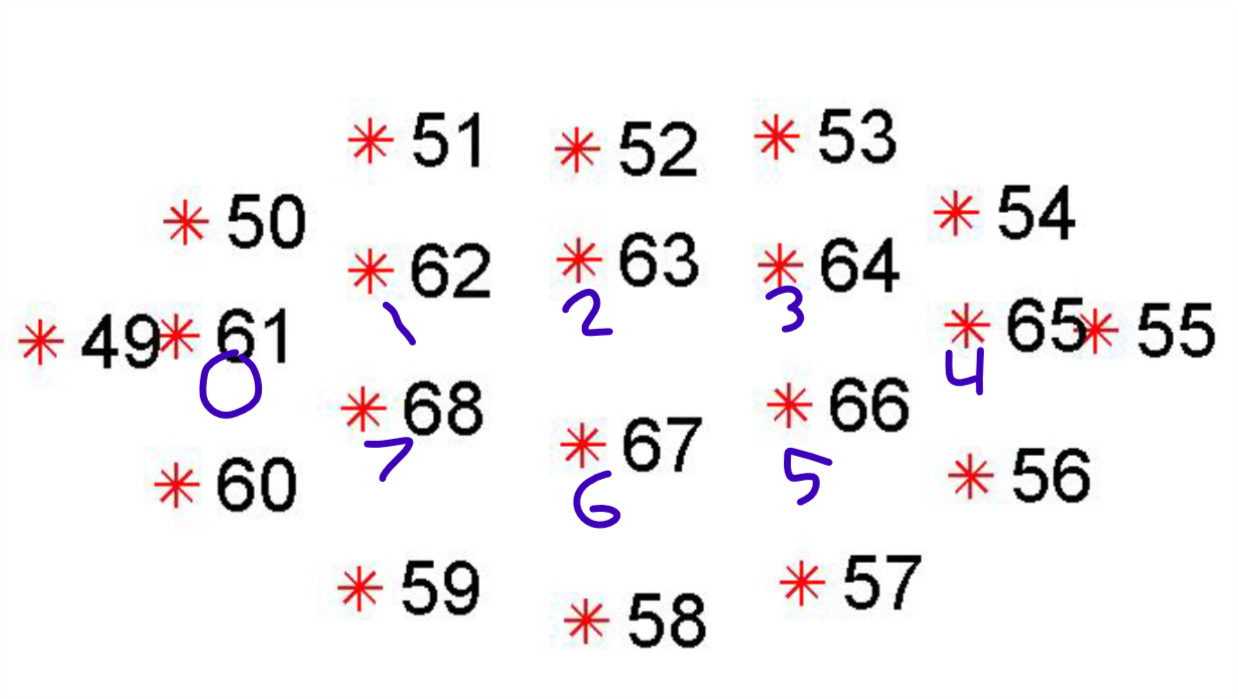


Figure : The mouth array

The function mouth\_aspect\_ratio works similarly to the eye\_aspect\_ratio function which is used to detect eye blinks. I thought if the eye function can be used to check to see if and eye is opening or closing, what if i used the same function but with some tweaks. Based on the following picture:

|  |  |
| --- | --- |
| Eye landmarks | Mouth landmarks |
| Figure : Eye landmarks | Figure : Mouth landmarks |

The following code is the mouth\_aspect\_ratio function:  
 def mouth\_aspect\_ratio(mouth):  
 # compute the euclidean distances between the two sets of  
 # vertical mouth landmarks (x, y)-coordinates  
 A = dist.euclidean(mouth[1], mouth[7])  
 B = dist.euclidean(mouth[2], mouth[6])  
 C = dist.euclidean(mouth[3], mouth[5])

# compute the euclidean distance between the horizontal  
 # mouth landmark (x, y)-coordinates   
 D = dist.euclidean(mouth[0], mouth[4])  
   
 # compute the eye aspect ratio  
 mar = (A + B + C) / (3.0 \* D)  
   
 # return the mouth aspect ratio  
 return mar

We first determine the distance of the vertical landmarks. Comparing 62 to 68, 63 to 67 and 64 to 66. We then determine the distance of the horizontal landmarks. Comparing 61 to 65.

We then used the same eye aspect ratio function we used before but with some changes. We added another vertical landmark C and multiplied D by three instead of two because we are using three vertical landmarks.

1. Now that we have the aspect ratio for the mouth, we can now set some variables.

**MOUTH\_AR\_THRESH = 0.09**: this will be used for how much distance the mouth has opened. With testing we have found that 0.09 was a good value for detecting a mouth opening, results may vary.

**MOUTHCOUNTER = 0:** used to cou

**MOUTHTOTAL = 0**: used to contain the total amount the mouth has opened for.

We can use these variables for checking certain conditions.

The following conditions were used for detecting a mouth opening:

if mouthMAR < MOUTH\_AR\_THRESH:

MOUTHCOUNTER += 1

**MOUTH\_AR\_THRESH:** is set to 0.9 and is used as the threshold for the mouth.

The first condition checks to see if the current ratio is less than the threshold that we set (0.09) if this this happens the mouth counter will be increased by one.

Else if MOUTHCOUNTER >= EYE\_AR\_CONSEC\_FRAMES

MOUTHTOTAL += 1

MOUTHCOUNTER = 0

**EYE\_AR\_CONSEC\_FRAMES:** is set to 2, it used to see how many successive frames that have a mouth ratio less the threshold.

The next condition checks to see if the mouthCounter is greater or equal to the eye\_ar\_consec\_frames if this happens mouthTotal will be increased by 1, indicating a mouth opening. It then resets the mouthCounter back to zero so we can check for another mouth opening.

1. Once we have obtained the mouthTotal, we will the following code to display the value of it on the frame to show the user how many times they have opened their mouth.

cv2.putText(frame, "Mouth Opened: {}".format(MOUTHTOTAL), (10, 330),  
cv2.FONT\_HERSHEY\_TRIPLEX, 0.7, (0, 204, 0), 2)

The put text function is used to display text on the frame. We used this function to display Mouth Opened: MouthTotal (how many times the mouth has been opened).

## **Task**: Recognize speech input from the microphone

**Author:** Aman Multani / Maryum Awan

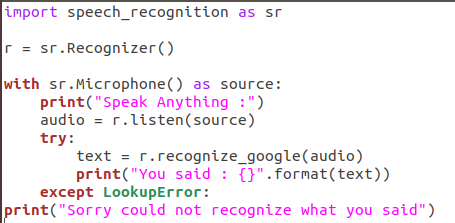


Figure : Speech\_recognition code

**Solution Detail:**

1. Using the library for performing the speech recognition with support of several engines and API’s online and as well as offline.
2. Obtain audio from the microphone
3. Use the default microphone as the audio source
4. Listen for the first phrase and extract it into audio data
5. Recognize speech using Google Speech Recognition
6. Shows error if speech is unintelligible

## **Task:** Detecting and retrieving the audio during the video stream

**Author:** Aman Multani

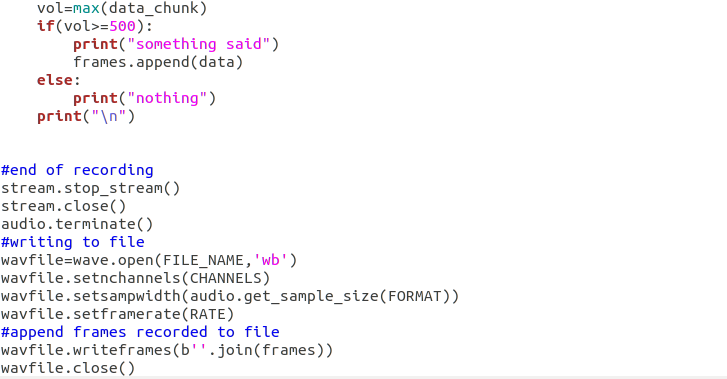
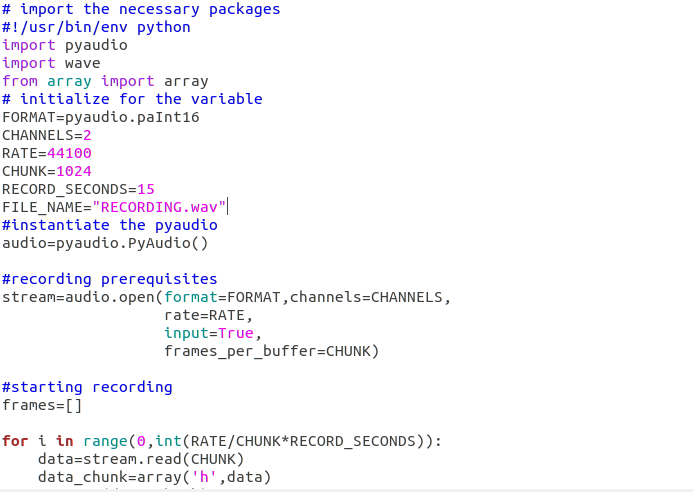


Figure : Speech recording

**Solution Outline:**

1. The speech recording and saving into a wav format to analyze the data.
2. In this we are using PyAudio source and build it for your system. Be sure to install the port audio library development package (portaudio19-dev) and the python development package (python-all-dev) beforehand.
3. Also Using the [wave](https://docs.python.org/2/library/wave.html#module-wave) module which provides a convenient interface to the WAV sound format.

**Solution Detail:**

1. “RATE” is the sampling rate like a numbers of the frames per second.
2. “CHUNK” is the number of frames in the buffer which are splits into small signals.
3. Each frame will have 2 samples and “CHANNELS=2”. using calculated using the function: pyaudio.get\_sample\_size(pyaudio.paInt16).
4. Size of the each sample is 2 bytes and frame is 4 bytes.
5. In the "FRAMES" list, size of each element must be 1024\*4 bytes, for example, size of frames[0] must be 4096 bytes. However, sys.getsizeof(frames[0]) returns 4133, but len(frames[0]) returns 4096.
6. “For” loop executes “int (RATE / CHUNK \* RECORD\_SECONDS)” times. RATE \* RECORD\_SECONDS is the number of frames that should be recorded. Since the for loop is not repeated for each frame but only for each CHUNK. And the loop is divided by the number of CHUNK.
7. And last but not least if we want to see the hexadecimal values , maybe we can try [hex(x) for x in the frames[0]].we can get 2 byte number use the format string’<H’ with the struct module.

## **Task:** Detecting and retrieving the audio during the video stream

**Author:** Ahmad Kaafi

**Solution Detail:**

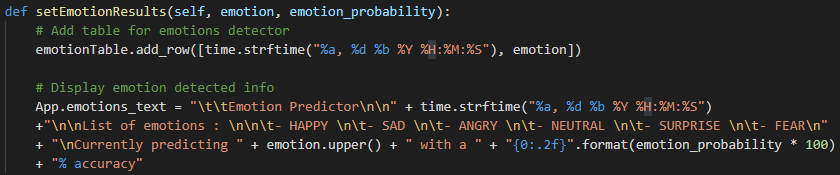
1. To display the results, we have to create an instance of our App class by doing the following

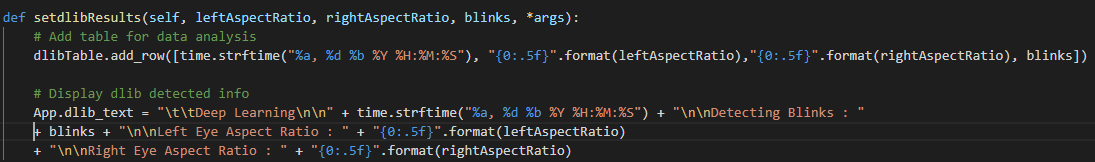
https://lh4.googleusercontent.com/BVnGksgh9SAINA04lnl4ykT9tcl2l2_nHht1mm0_6jE3S_r1k1PNQ6vg6lT0cxdQgEU98zdbVEUlPQMhOAyPfnn1NZK2cRr-GJUy-S6QcQRZUjmlGsVF6taajIjc0i2sDTI0alG7

1. Once we have got a result from either the facial capture or emotion capture classes, we will send the results to our App class

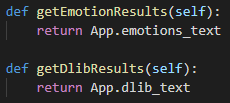
https://lh4.googleusercontent.com/kdb_vcFP5viDwBPVutpLlKeChfwo4Y3ELdn-6_bPWaqK-5aAaN706js-6uEdvCKRHGG6hw5_vaZ_p2Njqox27ANikKS_NXJw8XITbdHDg0Eq49HU33xLr0zJ9kud2bAoVXCfqnbK

1. The set functions does the following, it gets the result so that we can properly format it the display.

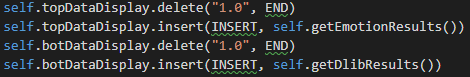




1. We than get the values we have set by using our get functions.



1. After we have finished obtaining our results, we will then display it to our text boxes by using the update method:



1. The top data display will have the emotions results and the bot data display will have our Dlib results displayed in real-time.

## **Task:** Creating the splash screen

**Author:** Maryum Awan



**Solution Detail:**

* This is a splash screen which appears on the front while the GUI is loading.
* It uses TTK Progress bar control for this.
* It shows splash screen in the center of the screen
* We gather the height, width and the center point of the Screen
* Using the geometry function the center of the screen is determined
* Create a progress bar that has desired countdown length.
* Attach progress bar under splash image
* Start discounting of the progress bar.
* Set window to topmost window so that no window can hide it.

## **Task:** Determining the color in front of the Webcam

**Author:** Aman Multani / Maryum Awan

**Solution Detail:**

1. Take each frame of the video.
2. Convert each frame from BGR to HSV color-space.
3. Threshold the HSV image for a range of blue color.

- Web camera captures the frames in every instance and is open during the entire course of the program.

- After capturing the live stream frame by frame, we are converting each frame in BGR color space to HSV color space.

- There are more than 150 color-space conversion methods available in OpenCV. But we will investigate only two which are most widely used ones, BGR to Gray and BGR to HSV.

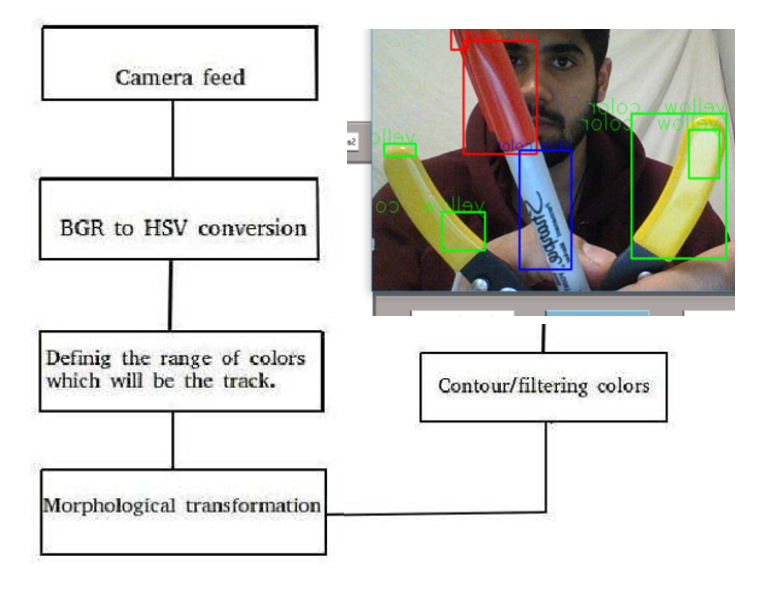
- For color conversion, we use the function cv2.cvtColor(input\_image, flag) where flag determines the type of conversion.

- For BGR to HSV, we use the flag cv2.COLOR\_BGR2HSV. Now we know how to convert BGR image to HSV, we can use this to extract a colored object. In HSV, it is easier to represent a color than RGB color-space.

We will do some Morphological transformation on the colors we’re going to track- Basically, this done to remove small noises in the image It is normally performed on binary images. It needs two inputs one is our original image second one is called structuring element or kernel which decides the nature of the operation.

the next part is just to contour the following colors region We’ve previously described. Contours can be described as a curve joining all the continuous points along the boundary having the same color or intensity. There are three arguments in cv2.findContours function first one is source image second is contour retrieval mode third is contour approximation method

We then display the frame, res on the separate windows using the imshow function.



## **Task:** Ability to record the video stream with the full functionality of different function

**Author:** Aman Multani / Maryum Awan

**Solution Detail:**

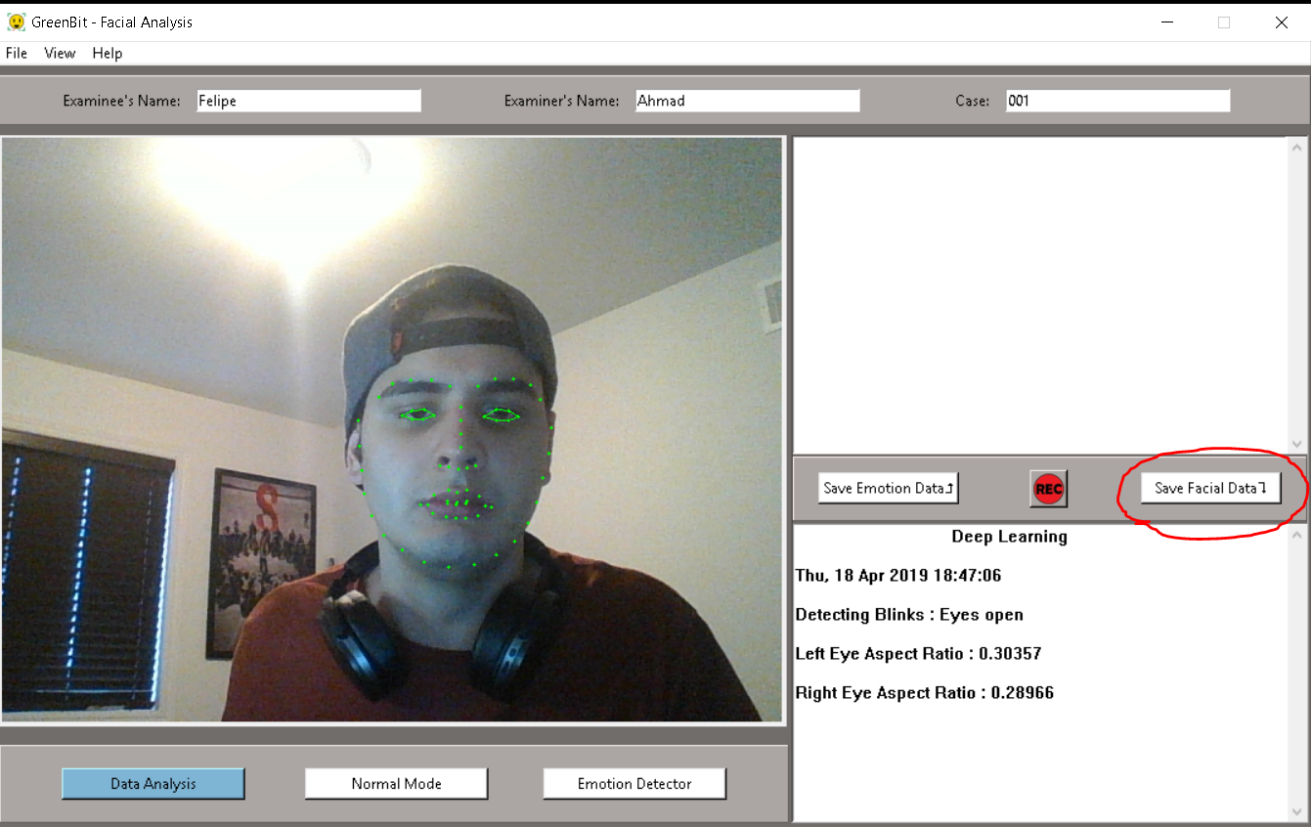
* We are using the global variable for the boolean expression to set the is\_recording for true while the record button is clicked
* Then we gather each frame from the video\_source
* Then we define the codec and create VideoWriter object.
* With the help of the VideoWriter the output is stored in 'output.avi' file.

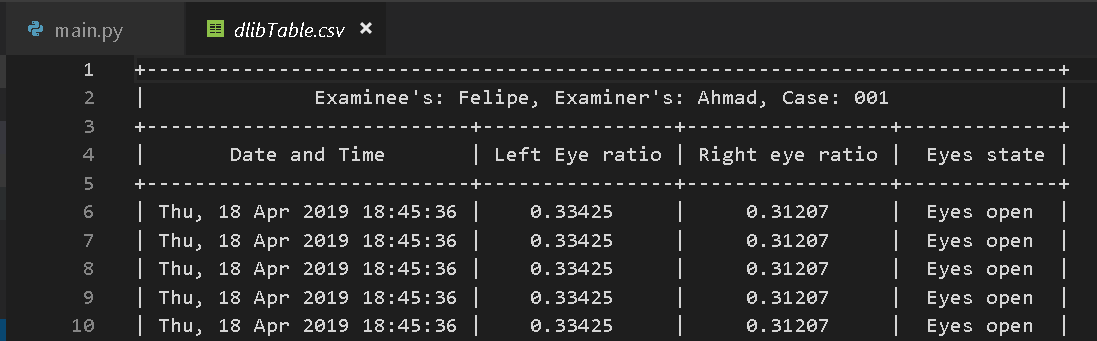
## **Task:** Save Facial Analyses Data

**Author:** Felipe de Paula

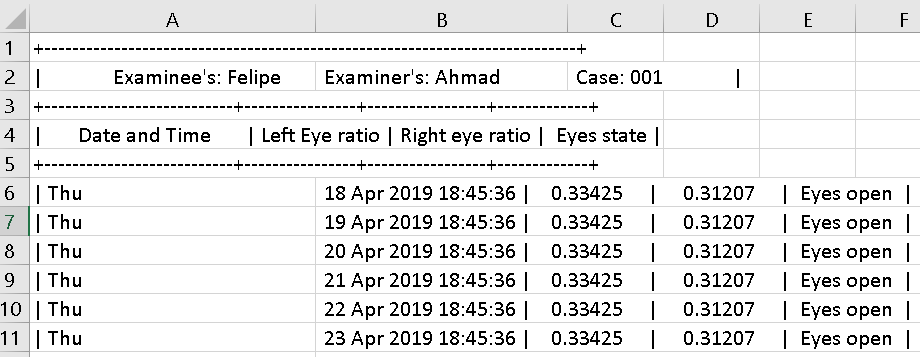
**Solution Detail:**

* User clicks on save emotion data or save facial data button
* Data is saved into CSV (comma separated values) file format to be opened and viewed on Excel





Facial analysis result printed on Visual Studio code



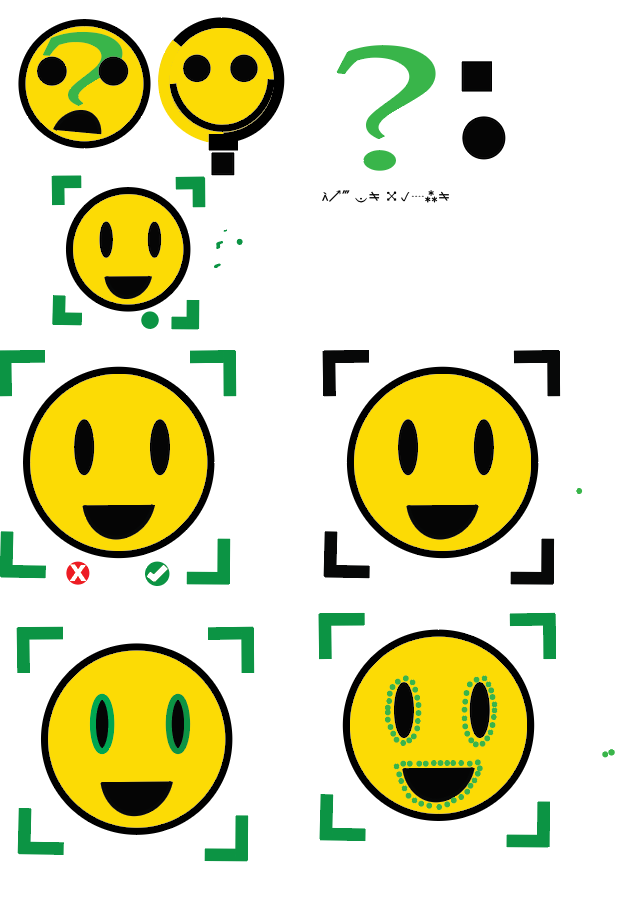
Facial analysis result printed on Excel

## **Task:** Design Application Log

**Author:** Felipe de Paula

**Solution Detail:**

* Use Adobe illustrator to design examples of facial analyses icons
* Select best logo based on group members opinion
* Convert PNG image format into ICO image format
* Implement Logo into the application

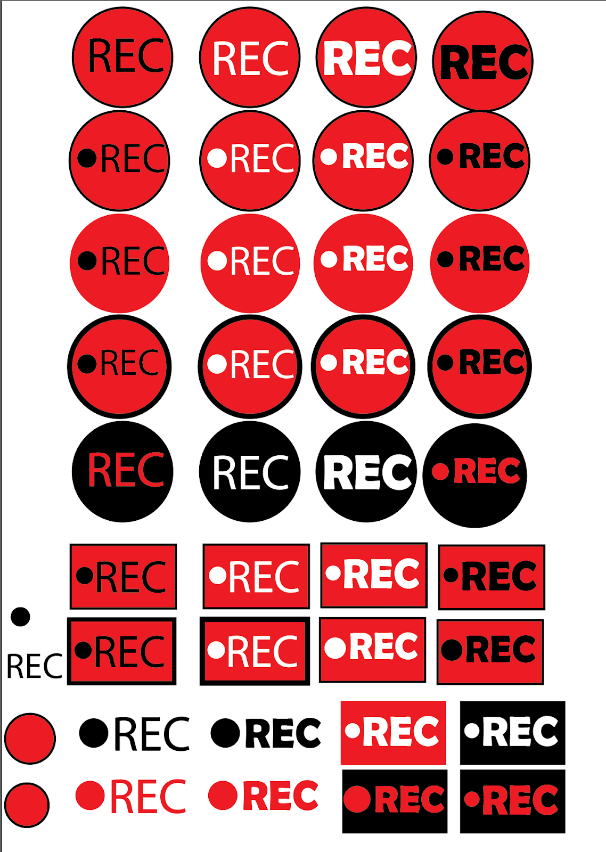
Logo prototypes                                  Final version logo

## **Task:** Design

**Author:** Felipe de Paula

**Solution Detail:**

* Use Adobe illustrator to design examples of record button logos
* Select best logo based on group members opinion
* Convert PNG image format into ICO image format
* Implement Logo into the application

Record button prototype                             Final version for the record button

## **Task:** Design UI

**Author:** Nicolas Daigle

**Solution Detail:**

* **Menu Bar**
  + File
    - Open
    - Save data
    - Start Recording
    - Stop Recording
    - Exit
  + View
    - Change Views
      * Normal View
      * Facial Data
      * Emotion Detection
  + Help
    - About
* **Scene/Layout**
  + Top frame for metadata entered by the user
  + Left side, The video stream output
  + Right side, Vertical stack of text boxes, the top text box for emotion data, middle frame containing the save and record buttons and bottom text box for facial data.
  + Bottom frame for view control buttons
* **Other features**
  + Custom popup window when saving data from the menu bar or using its key bind
  + On Screen visual to notify users when they are recording
  + Visual color change indicating which video processing mode is running
  + Buttons and menu bar options have key binds/shortcuts

Convert PNG image format into ICO image

# Lessons learned

Pros

* As a team we learned that we work well when we are together as a group

Cons

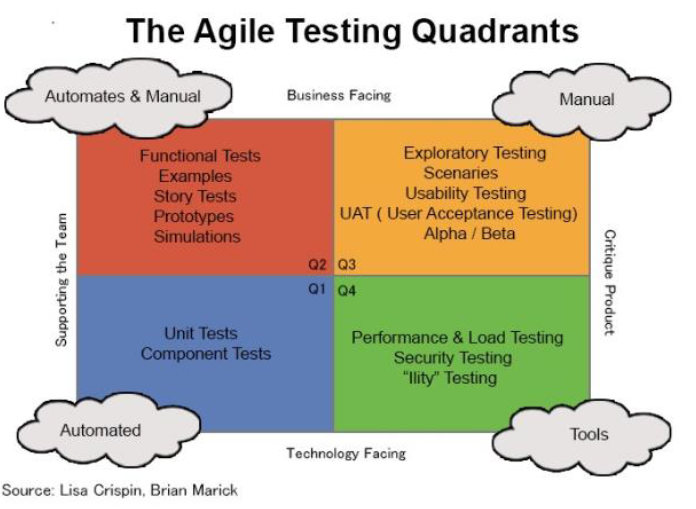
* Communication is lack luster and could be improved upon in the next semester

Plans for next semester

1. To work more on Audio for detecting the wavelength of the sound and the pitch etc. and compare it with when a person is angry or how many times a person took a pause while speaking.
2. Work on the thermal and threshold frequencies of human body to detect the body temperatures, by comparing the average thermal value of human with difference with high and low.
3. For detecting the thermal value, we might need the thermal camera for calibrating the thermal values of human body.
4. Regarding the datasets, we need to plan a way for us to obtain more data so that our lie detector can be more accurate.
5. To implement AI machine learning so that we can use our currently generated data to produce an accurate percentage that the user is lying or not.

# Test Plan

# We put a lot of focus into the red quadrant, as our main testing is done by simulations



# References

<https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>

<https://parade.com/57236/viannguyen/former-cia-officers-share-6-ways-to-tell-if-someones-lying/>

This website ^ referenced the book

SPY THE LIE: Former CIA Officers teach you how to detect deception.

<https://www.amazon.com/dp/1250029627->

<https://www.forensicscolleges.com/blog/resources/10-signs-someone-is-lying>

<https://people.csail.mit.edu/hubert/pyaudio/>