F.Y.D
FaceID | Yolo | Dataset Generation

By

YASH THESIA [15BCE126]
TIRTH PANDYA [15BCE127]

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

(I)

F.Y.D
FaceID | Yolo | Dataset Generation

Mini Project - II

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering

By

YASH THESIA [15BCE126]
TIRTH PANDYA [15BCE127]

Guided By
Prof. Anitha Modi
[DEPARTMENT OF COMPUTER ENGINEERING]

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

(II)

CERTIFICATE

This is to certify that the project entitled "FaceID, Yolo and Dataset Generation" submitted by YASH THESIA (15BCE126) & TIRTH PANDYA (15BCE127), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Prof. Anitha Modi
Assistant Professor
Department of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

Prof. and HOD Dr. Sanjay Garg
Dept. of Computer Engineering
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

In the light of this Mini-Project II entitled “FaceID, Yolo and Dataset Generation” , we would like to thank Prof. Anitha Modi who has been exceptionally positive and helpful in this course. We would also like to thank our friends and parents who have supported us and encouraged us for our Innovative Idea. Finally, we appreciate the course designers and management of the University for providing us the opportunity of “ Implementing our Technical Skills and Knowledge ” to build a project in our own field of “ Computer Engineering ”.

YASH THESIA [15BCE126]

TIRTH PANDYA [15BCE127]

ABSTRACT

F.Y.D or Face ID, Yolo and Dataset Generation is a fusion of next generation Computer Vision and Training paradigms.
It is an innovative Web Application that registers users and enables them to upload photos onto a server database. After collecting a fair amount of images from different users, the application runs different Object Detection and Face Recognition algorithms (Yaar Cascades | Yolo) in the background.
The Result of the detection and recognition are then displayed to the user, either as a static image or as a Real-Time Video.

The project is a fusion of three major components :

1. Dataset Generation.
2. Facial Recognition using Haar Cascades.
3. Object Detection using YOLO.

Our objective was to first be able to collect a generous amount of images as dataset in the server. Secondly, we had to apply the algorithms of Facial Feature extraction included in the Haar Cascades library of OpenCV.

Next, we trained the images using the Trainer Function, and then the Recognizer Function was used to Tell who the user is in real-time.

Finally, we enhanced both the components : The Web UI for Dataset Collection, and the Python Script for detection and recognition and fused them into a single application : The F.Y.D.

CONTENTS

List of Figures

Chapter 2:

Chapter 3:

Chapter 1
Introduction

1.1 **Overview** :

F.Y.D is a Web Application that enables users to upload images of faces and objects to create a database of dataset. Additionally, it provides state-of-the-art algorithms to detect objects and recognize faces of the users. It is a compact and portable application that can be used with almost any device, which has a camera which supports browsers.

In the current scenario with the advent of Machine Learning and Artificial Intelligence, applications like these are necessary to generate sufficient data to train the various machine learning models.

This application combines the concept of both dataset generation and facial recognition and identification, so in a way it satisfies both research and commercial aspects of a powerful and modern application.

1.2 **F.Y.D : The Web App** :

This app is supported in all the major browsers across all the devices. It's UI is very simple and intuitive. There is a registration and login page, followed by the main page with all the navigation options.

1.2.1 **Minimum Requirements :**

Device Requirements :

- Camera Functionality (atleast 2MP)
- Browser Support : IE 8 | IE 9 | IE 10 | Firefox | Opera 12 | Safari 6 | Chrome.
- Internet Connectivity.

Developer Tools :

- Python 3.5+
- HTML5 | CSS
- PHP | MySQL
- PyCharm : Python IDE

1.2.2 **Using the Web Application :** The app provides a very simple and easy to use UI. Provided below is the instruction set to use the O.S.S App :

- Go to the Website (here Localhost).
- Click on the Button " Login/Signup ".
- For new users, go to the signup page and register.
- After registration, go back to the login page and login.
- After successful login, the Main Navigation page of the website is displayed.
- Choose the Fetch and Train your data.
- Enter the username. The camera will start taking a series of 50 photos in sequence, which will detect your face and capture it, following the training of your faces in the background.
- After successful capture and train, select the 'Who am I ?' option to recognize your face in real-time.
- Select the 'Detect a Face Boundary' to use the Haar Cascades face detection algorithm to detect face Boundary in real-time.
- For Object detection, first capture the image using 'capture a photo' option. Download it after capture.
- Next, upload the photo to the server.
- Finally, Run YOLO and detect the objects.

1.3 **Objective of Study :**

This Project enables the aspirant to learn about facial recognition and object detection by the use of Libraries in Python for performing the same. It also develops the skill of programming in Web Designing and Python, and training with algorithmic approaches such as Haar Cascades and Yolo.

1.4 **Scope of Work :**

The course of this project requires strong understanding of the concept of Facial Feature Recognition and Object Feature extraction and its implementation using Libraries in Python.
Furthermore, the core idea of the project, i.e., generate dataset and recognise faces, requires good programming skills for working with matrices and good command over approaches for developing algorithms, especially Convolution Neural Networks.

Chapter 2

Literature Survey

2.1 **Dataset Generation :**

- The concept of Dataset Generation requires the user to upload a series of images.

- The images may be of various types, they may images of the user’s face, or any random everyday object.

- The images are labelled with the users’ unique ID - provided by their names.

- The labelled images are then stored in the database in the server.

2.2 **Face Recognition and Object Detection :**

2.2.1 **Haar Cascades :**

- Paul Viola and Michael Jones introduced the concept of Object Detection through Haar feature-based cascade classifiers.

- A crucial advancement in the field of OpenCV - Haar Cascades is a Machine Learning based approach. It uses a cascade function trained from a positive and negative images. The trained function is then applied to new images for detection of objects.

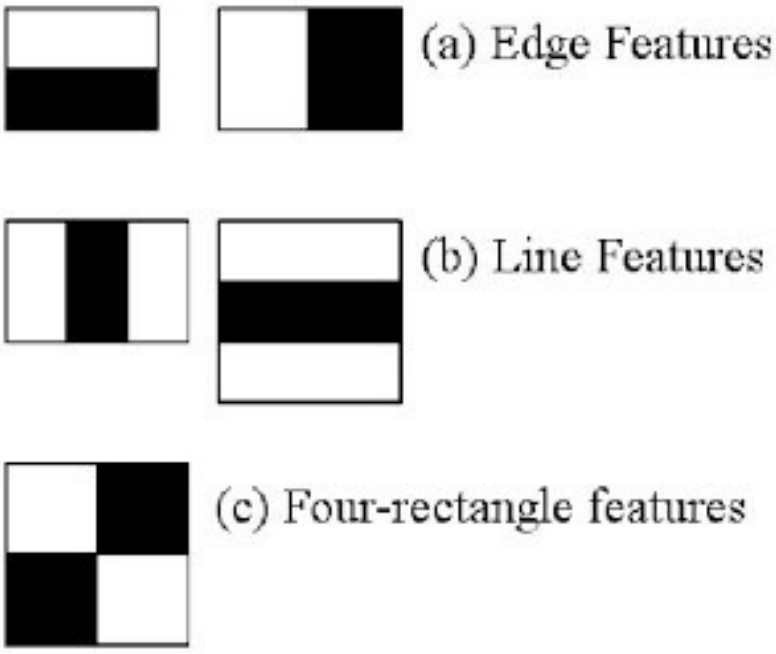


Fig 2.1 : Line Features

- So the first step is Face Detection. Using the Haar-features as depicted above, we try to detect such features. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

- Presently all conceivable sizes and areas of every piece is utilized to compute a lot of features. For each component figuring, we have to discover aggregate of pixels under white and dark rectangles. To understand this, they presented the integral images. It rearranges figuring of total of pixels, how extensive might be the quantity of pixels, to an operation including only four pixels. This makes the algorithm run very fast.

- Even though a large number of features are calculated, most of them are unnecessary. An example is shown below. Top column demonstrates two great features. The principal feature chosen appears to concentrate on the property that the district of the eyes is frequently darker than the area of the nose and cheeks. The second component chose depends on the property that the eyes are darker than the extension of the nose. Be that as it may, similar windows applying on cheeks or some other place is unessential. The task of selecting the best features out of the vast multitude is accomplished by Adaboost.

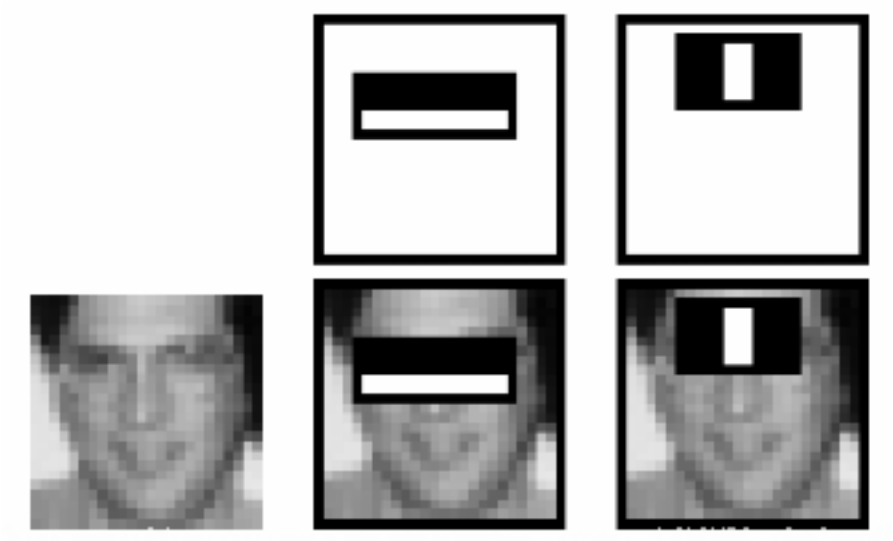


Fig 2.2 : Facial Line Mapping

- For this, we apply every single element on all the preparation images. For each element, it finds the best limit which will group the appearances to positive and negative. Still, there will be blunders or misclassifications. We select the features with least blunder rate, which implies they are the highlights that best characterizes the face and non-confront images.

- Each image is given an equivalent weight to start with. After every arrangement, weights of misclassified images are increased. On the other hand same process is finished. New mistake rates are computed. Likewise new weights. The procedure is proceeded until the point that required precision is accomplished or required number of features are found.

- Last classifier is a weighted total of these component classifiers.The accuracy is of 95% can be achieved with only 200 features.

- Ultimately, Cascade of Classifiers was presented. Rather than applying all the 6000 highlights on a window, gather the highlights into various phases of classifiers and apply one-by-one.

- LOCAL BINARY PATTERN HISTOGRAM (LBPH) TECHNIQUE

    - In this algorithm, pixels are represented as vectors of data points in HD vector space. HD vector space can be vast and nebulous, so we use a sub-space which is lower in dimensions.

    - LBPH considers texture descriptor which is useful to symbolize Faces. Because face data can be split as compositions of patterns of micro textures. Basically LBPH is carried out in 3 stages they are :

        1. Feature extraction,
        2. Matching,
        3. Classification

    - First, the image is captured and converted into Grey-Scale. Secondly, the features of haar are identified and it is decided whether there is a face or not. Finally, it is mapped to a face in case of a face being detected.

    - The efficiency is 72%, and tilting of face upto 45 degrees is allowed.

2.2.2 **Yolo :**

- Earlier recognition systems repurpose classifiers or localizers to perform identification. They apply the model to an image at multiple areas and scales. High scoring districts of the image are considered identifications.

- YOLO utilizes an entirely unexpected approach. It applies a solitary neural network to the full image. This network partitions the image into locales and predicts bouncing boxes and probabilities for every district. These bouncing boxes are weighted by the anticipated probabilities.

- The model has a few points of interest over classifier-based systems. It takes a gander at the entire image at test time so its forecasts are educated by worldwide setting in the image. It additionally makes forecasts with a solitary network assessment not at all like systems like R-CNN which require thousands for a solitary image. This makes it to a great degree quick, more than 1000x quicker than R-CNN and 100x speedier than Fast R-CNN. See our paper for more subtle elements on the full framework.

- YOLO adopts a totally unique strategy. It's not a conventional classifier that is repurposed to be a protest finder. YOLO really takes a gander at the image just once (henceforth its name: You Only Look Once) yet astutely.

- Each of these cells is in charge of foreseeing 5 bouncing boxes. A jumping box portrays the rectangle that encases a question.

- YOLO likewise yields a certainty score that discloses to us how certain it is that the anticipated bouncing box really encases some question. This score doesn't say anything in regards to what sort of protest is in the case, just if the state of the crate is any great.

- For each jumping box, the cell likewise predicts a class. This works simply like a classifier: it gives a likelihood conveyance over all the conceivable classes. The rendition of YOLO we're utilizing is prepared on the PASCAL VOC dataset, which can distinguish 20 unique classes, for example,

    - bicycle
    - boat
    - car
    - cat
    - dog
    - person

- The certainty score for the jumping box and the class forecast are consolidated into one last score that discloses to us the likelihood this bouncing box contains a particular kind of question.

- Since there are 13×13 = 169 network cells and every cell predicts 5 bouncing boxes, we wind up with 845 jumping encloses add up to. Things being what they are the greater part of these cases will have low certainty scores, so we just keep the crates whose last score is at least 30% (you can change this limit contingent upon how precise you need the locator to be).

- The Neural Network architecture of YOLO is simple, it's just a convolutional neural network:

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

- This neural system just uses standard layer sorts: convolution with a 3×3 piece and max-pooling with a 2×2 bit. There is no completely associated layer in YOLOv2.

- The last convolutional layer has a 1×1 bit and exists to lessen the information to the shape 13×13×125. This 13×13 should look natural: that is the measure of the network that the picture gets isolated into.

- So we wind up with 125 channels for each matrix cell. These 125 numbers contain the information for the jumping boxes and the class forecasts. All things considered, every lattice cell predicts 5 bouncing boxes and a jumping box is portrayed by 25 information components:

    - x, y, width, height for the bounding box's rectangle
    - the confidence score
    - the probability distribution over the 20 classes

- The simplicity of YOLO lies in the fact that we just have to provide an input image, which is resized to 416x416 pixels, which runs through the CNN in one go, and gives a tensor (13x13x125) which describes the bounding boundaries for the grid cells as the output.

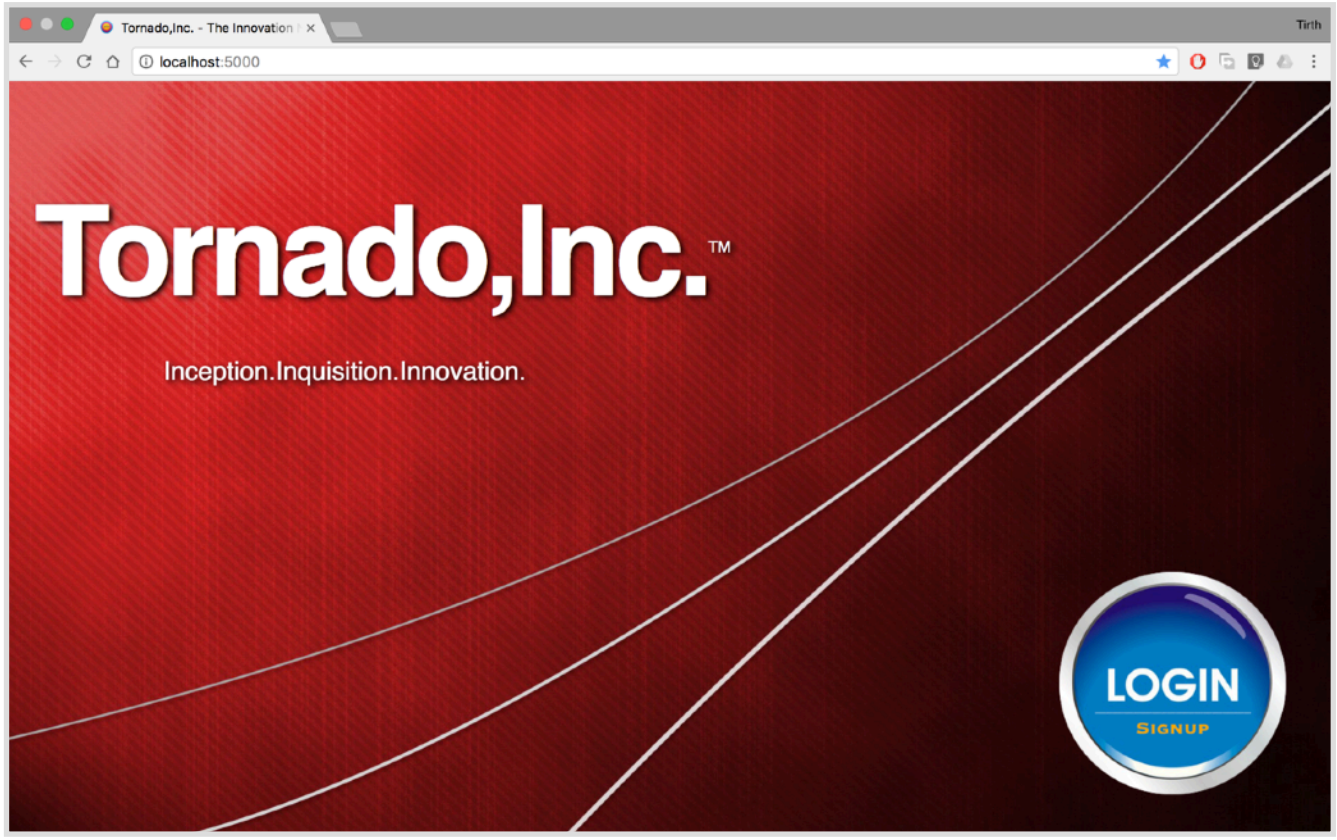
- Finally, all those boxes scoring less than 30% are discarded.

Chapter 3
Testing the Project

3.1 **Web Application Walkthrough**

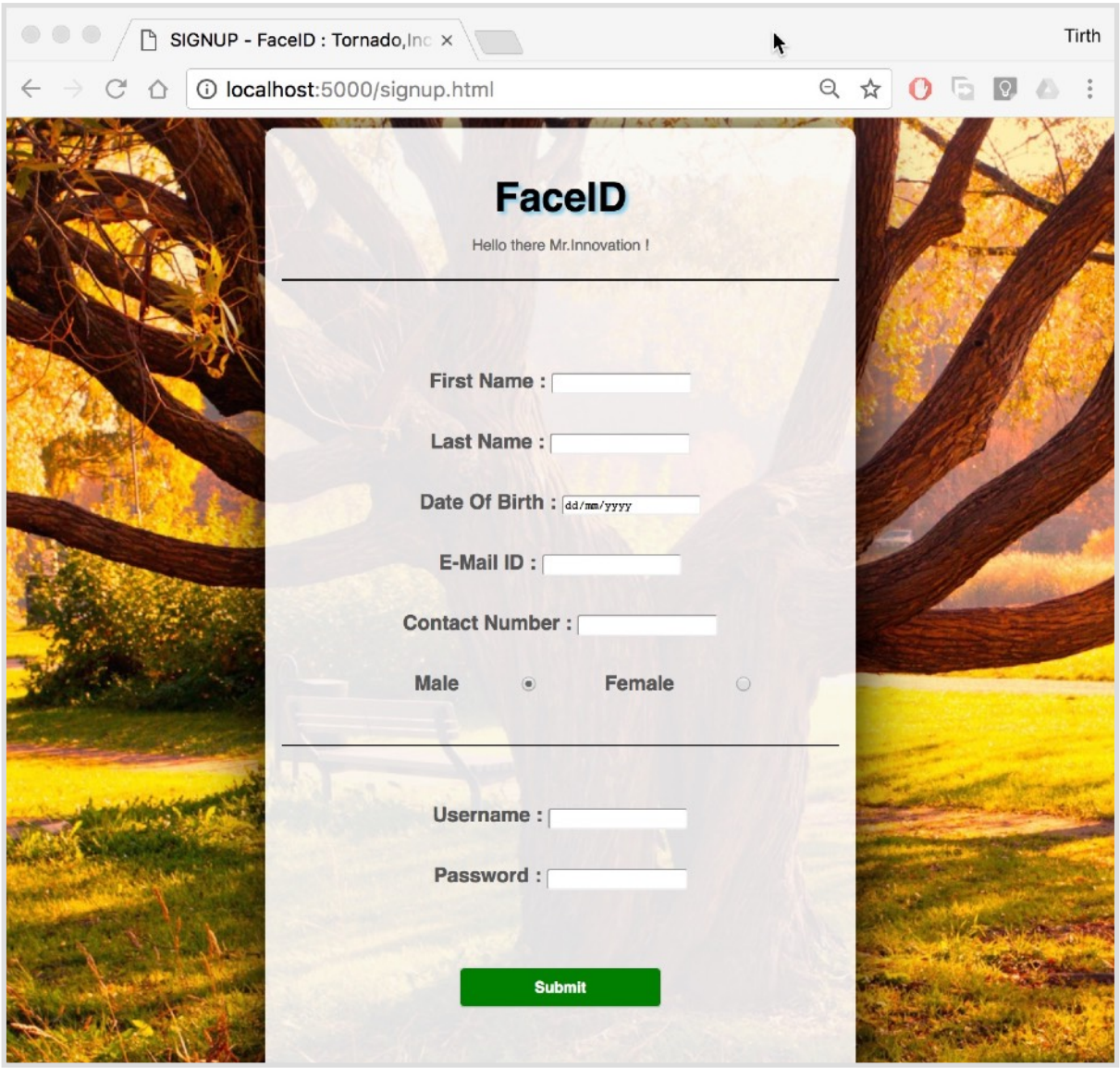
1. **Intro Screen :** The Welcome Screen of the Website, with the Company name and the Login Button :

Fig 3.1 : Intro Screen



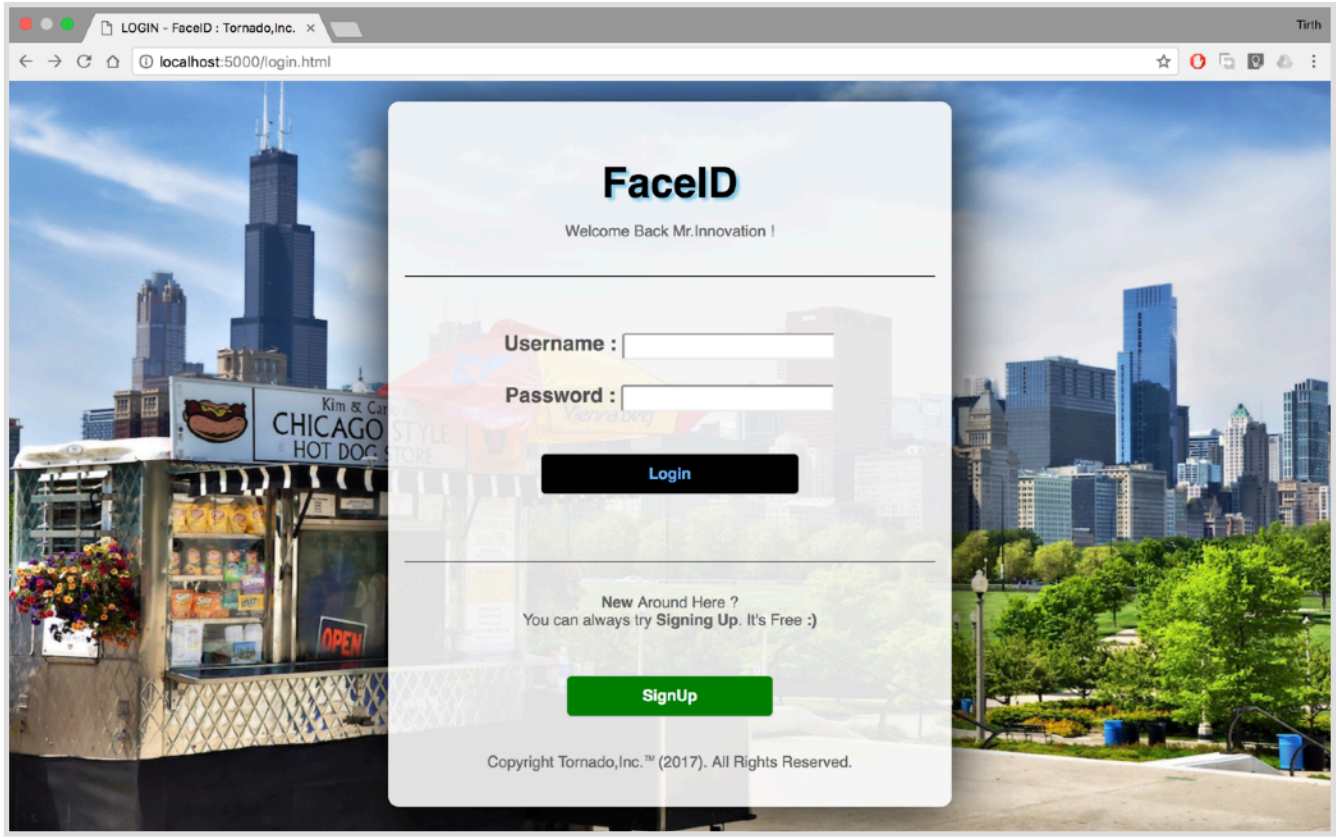
2. **SignUp Page :** The Registration / SignUp page is where the new users provide the necessary details to create a new account.

Fig 3.2 : SignUp Page



3. **Login Page :** The Login Page where the existing users can log into the application, where the underlying PHP code verifies the user by cross-checking with the Database.

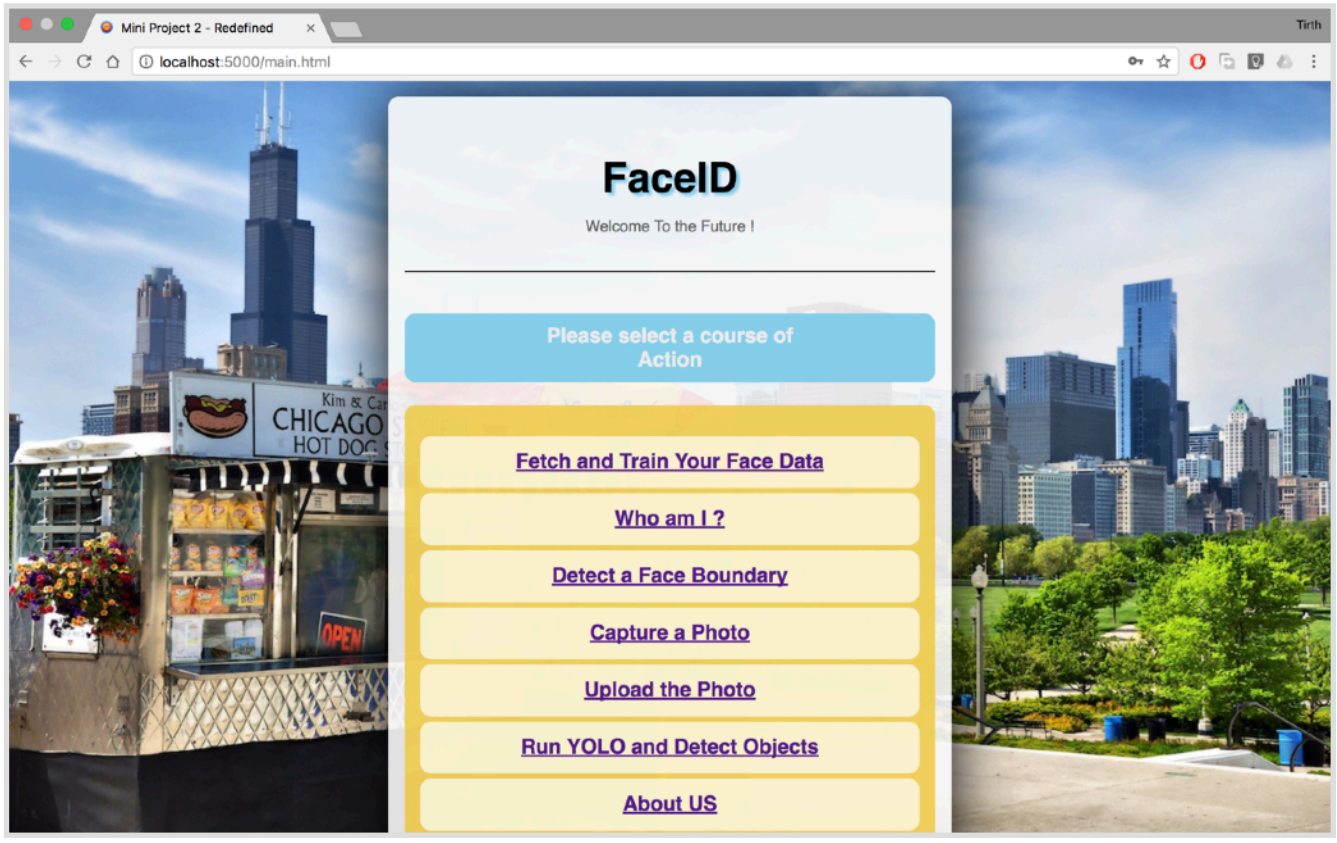
Fig 3.3 : Login Page



4. **Home / Main Menu Screen :** The Main navigation screen with all the action courses of the application :

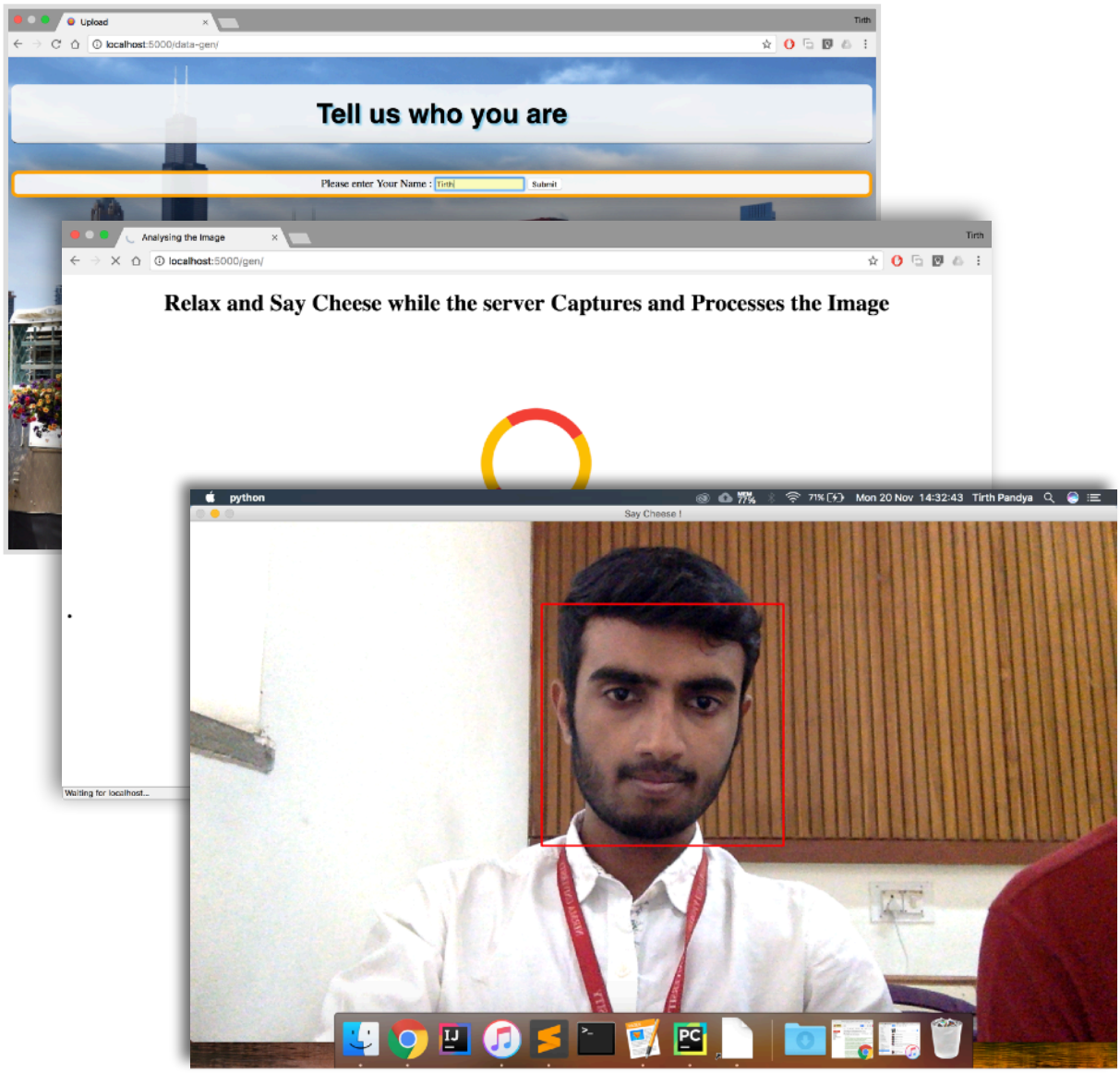
- Fetch and Train your face Data.
- Who am I ?
- Detect a Face Boundary.
- Capture a Photo.
- Upload a Photo.
- Run Yolo and Detect Objects.

Fig 3.4 : Home / Main Menu Screen



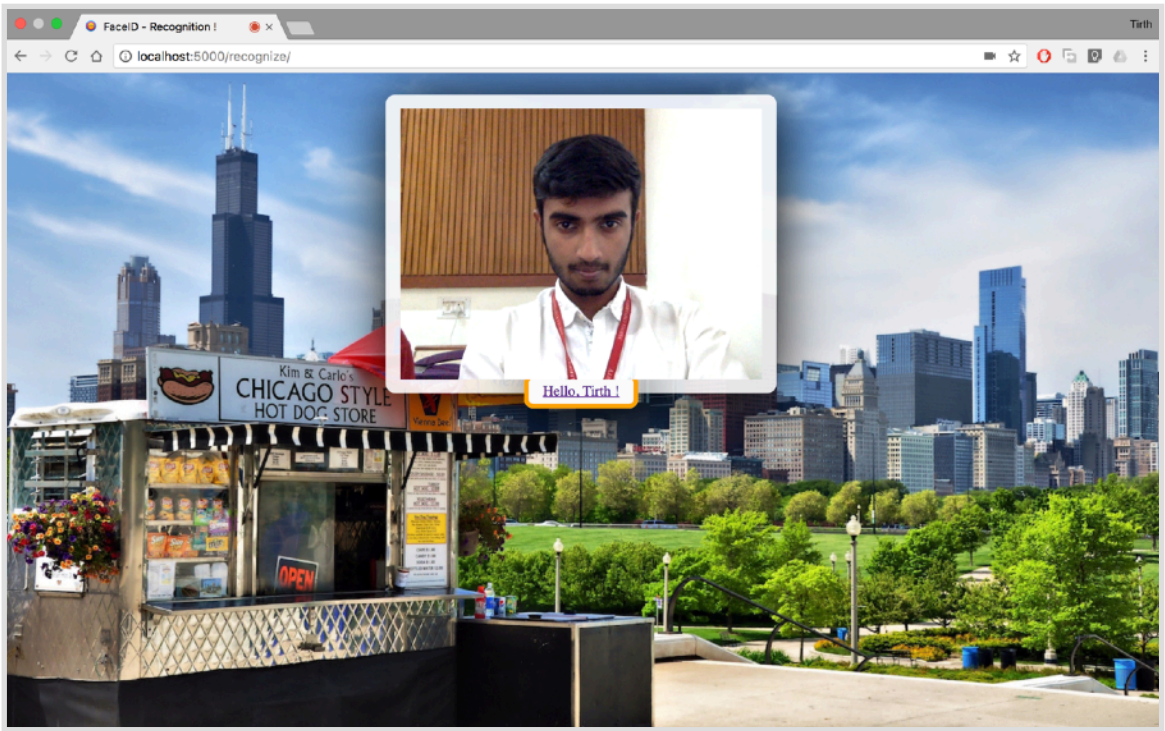
5. **Face Data Capture and Training Screen :** This page asks the user name to label the faces. It then captures a series of about 20 pictures of the user's face. Then it applies the haar cascades Lbph face recognizer classifier to train the captured images in the background.

Fig 3.5 : Face Capture and Training Screen



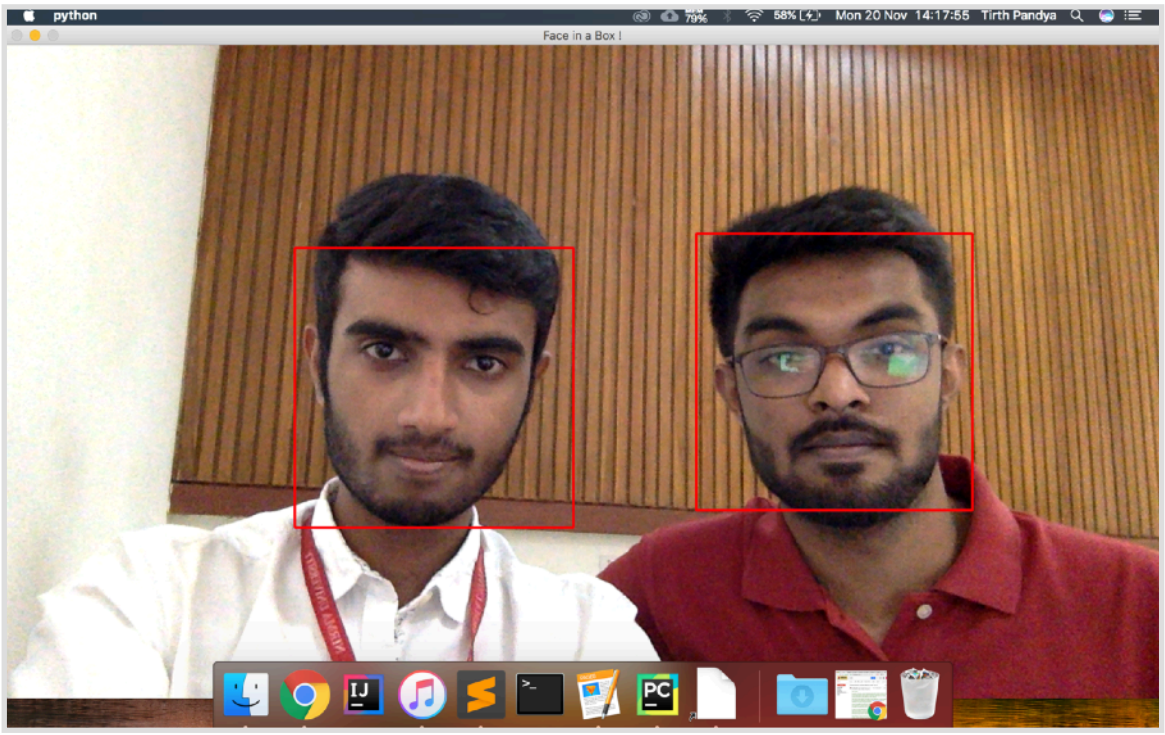
6. **Who am I ? The Face Recognition Screen :** This is the highlight of the application. Here the Face Recognizer function of Haar Cascades captures the real-time stream from the webcam and tries to identify the user in the frame. It then displays a “Hello, <user>” message with the predicted user name.

Fig 3.6 : Face Recognition



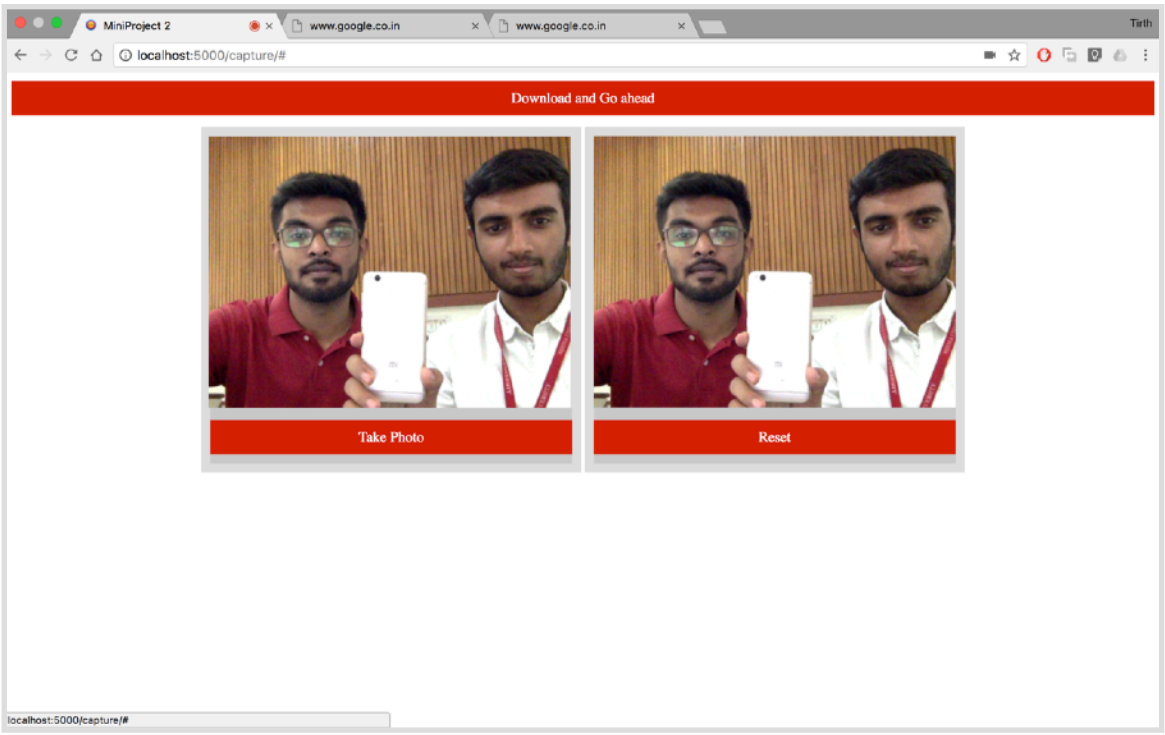
7. **Face Boundary Detection Screen :** The Haar cascades face detection algorithm runs in the background while a face boundary is drawn about the faces of multiple users in the live stream frame.

Fig 3.7 : Face Boundary Detection



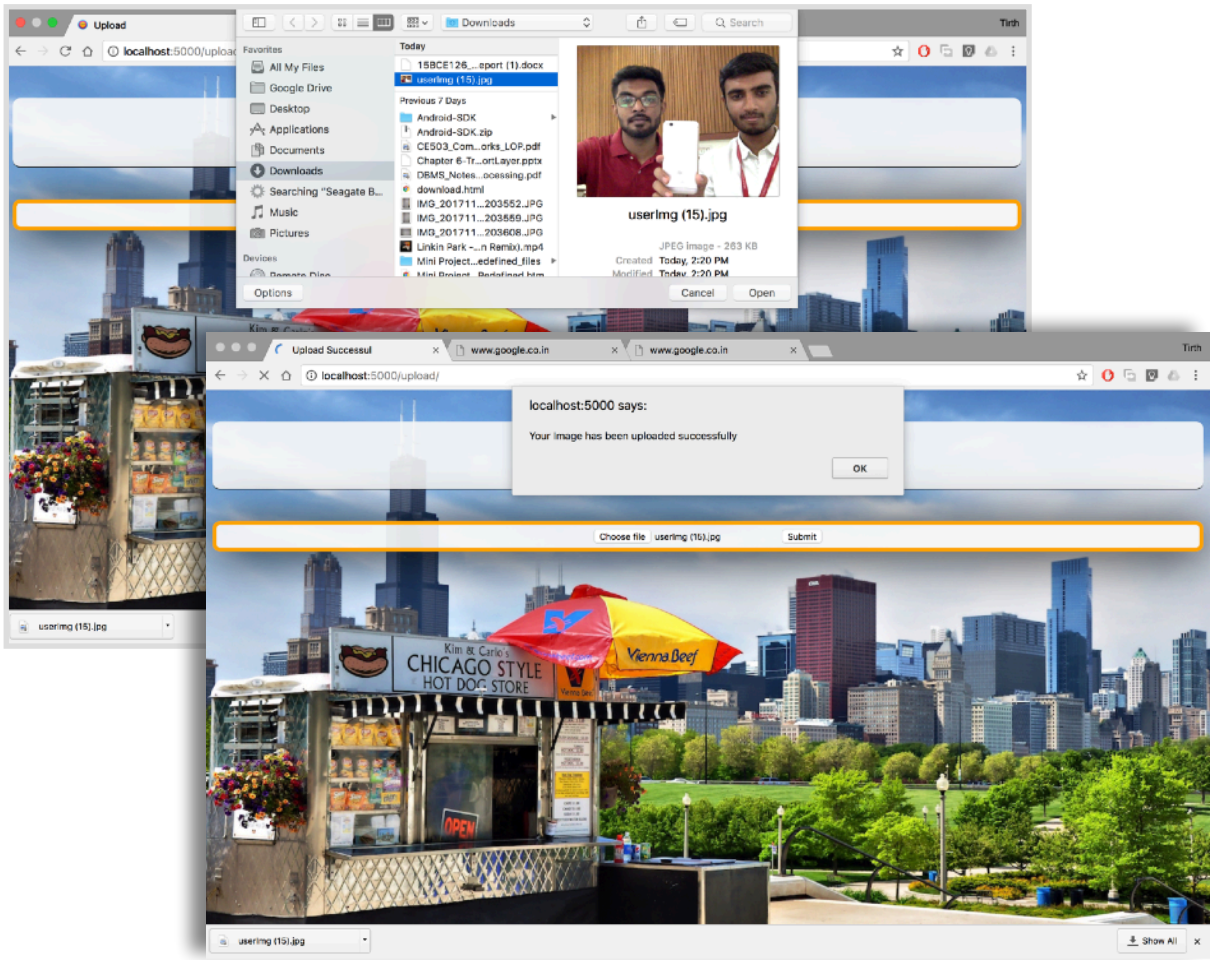
8. **Capture a Photo screen :** This screen allows the user to capture an image from the webcam livestream and upload it to the database later.

Fig 3.8 : Image Capture



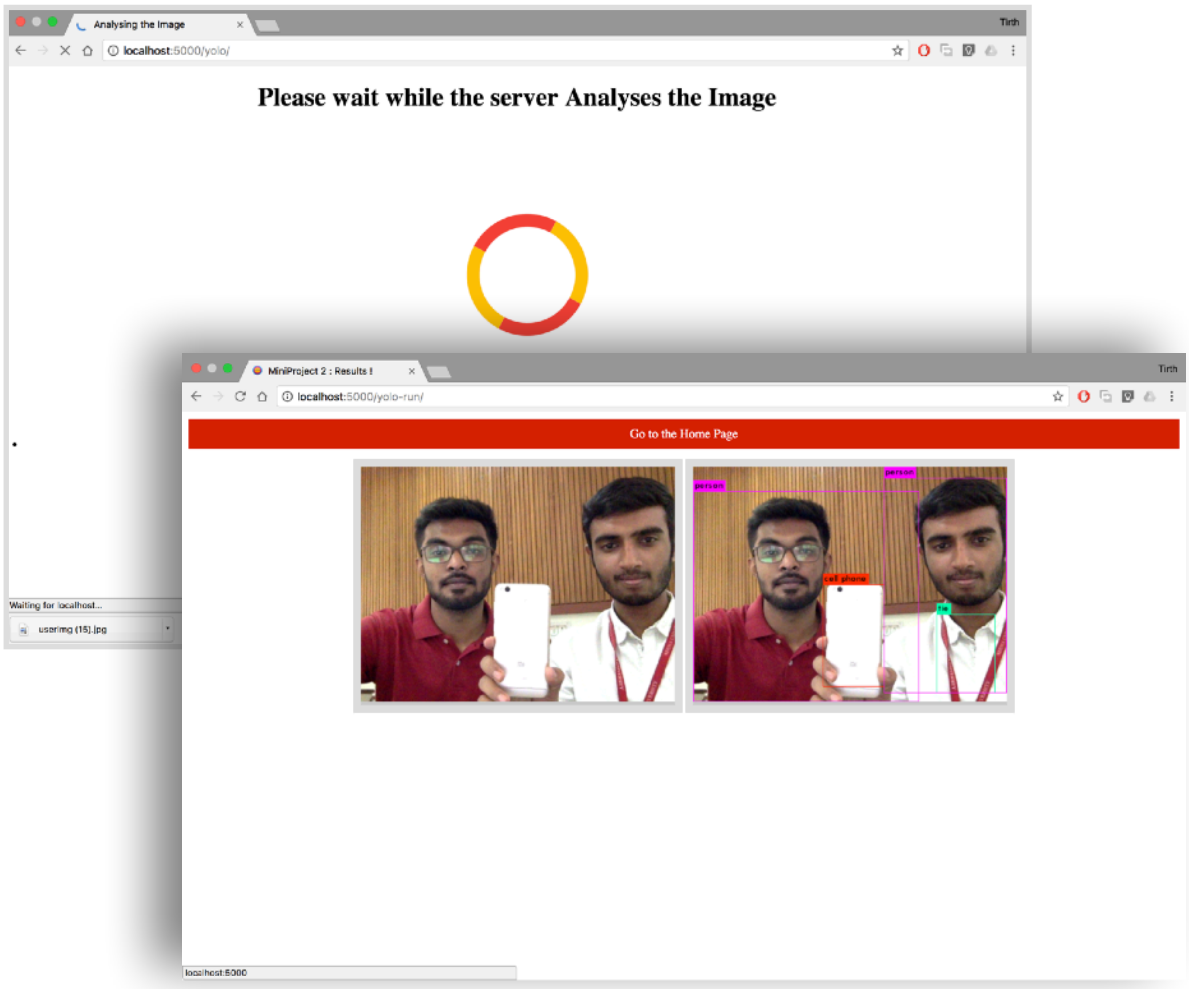
9. **Upload Photo Screen :** This screen allows the user to upload an image from the client system to the server database.

Fig 3.9 : Image Upload



10. **Run YOLO and Detect Objects Screen :** This screen is where the object detection takes place in the background by using YOLO. The YOLO algorithm runs it's analysis on the uploaded photo. The results are displayed with object boundaries and annotation.

Fig 3.10 : YOLO and Object Detection



4.2 **Troubleshooting**

- **THE APPLICATION HANGS UP** : Depending on the server-line capacity and hardware efficacy of the client, the application may freeze for sometime. Wait for a couple of minutes, else restart the browser.

- **IMAGE NOT RECOGNISED** : The captured image or the livestream video may not have the optimum lighting conditions required for detection and recognition. Try to get better lighting conditions, or a more powerful camera device for input.

- **PREDICTED USER NAME AND OBJECTS NOT ACCURATE** : The captured image may be skewed or angled. Capture the image again or select a straight facing one from the directory, and train again.

- **THE APPLICATION STOPS RESPONDING** : The application might have encountered a fatal error, or may have ended up in an exceptional case. Restart the application / browser or contact the developer if problem still persists.

Chapter F
Summary and Conclusion

F.1 **Summary**

To sum up about the project, following are the brief points that describe the project as discussed so far :

- The Project is entitled “F.Y.D” - FaceID | Yolo | DataSet Generation.
- It is a website application that captures / acquires Images from the users and maintains a DataSet of all such images.
- For face recognition, it trains on the face data collected and labels them with their respective user names.
- It then recognizes the user in the real-time using the trained weights by the HaarCascades Lbph algorithm, and displays the user-name predicted for the current user video stream.
- For object detection, it captures an image from the user that contains the object to be detected.
- It then runs the YOLO algorithm to detect objects using the pre-trained weights and annotates them.
- The resulting image is displayed alongside the original image in the final screen.

F.2 **Conclusion**

Project ‘F.Y.D’ has been an interesting and inspirational work for us, since it was ‘INNOVATIVE’ yet ‘CHALLENGING’ to implement. From the beginning of the journey with the documentation, to the comprehensive algorithms and approaches to extract the FACIAL FEATURES and detect OBJECTS, we really learnt a lot along the way.

The Web App itself turned out to be robust and it works quite well, exceeding our expectations, frankly speaking. This Web App is an example of several other ideas that are waiting to come out of the ‘BOX’.

It has definitely taught us the skills of ‘Team Work’ and ‘Co-Ordination’ to work on a project, giving us a sort of ‘HANDS-ON’ experience of the Industrial Field Work, along with its own set of obstructions and difficulties. But we are more than delighted to have faced them positively and more importantly, overcome a majority of them happily.

In the end, we would like to thank all those who are directly or indirectly a part of this project, the Mini-Project II titled:
‘ F.Y.D ’.

References

1. Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001
2. Papageorgiou, Oren and Poggio, "A general framework for object detection", International Conference on Computer Vision, 1998.
3. Face Recognition using Local Binary Patterns (LBP), By Md. Abdur Rahim, Md. Najmul Hossain, Tanzillah Wahid & Md. Shafiul Azam
4. You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon University of Washington pjreddie@cs.washington.edu, Santosh Divvala Allen Institute for Artificial Intelligence santoshd@allenai.org.

Appendix - A : List of Useful Websites

1. OpenCV Haar Cascades, https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html/
2. Real-Time Object Detection Algorithm (YOLO) , https://pjreddie.com/darknet/yolo/
3. Face Recognition OpenCV - Training A Face Recognizer - The Codacus, https://thecodacus.com › Computer Vision › Face Recognition/
4. "PyCharm IDE" https://www.jetbrains.com/pycharm/
5. "Practical Python and OpenCV", https://www.pyimagesearch.com/