*A*

***Practical Training Seminar Report***

*on*

**Computer Vision and Machine Learning**

taken at

**Adhoc Networks, Jaipur**

*submitted to*

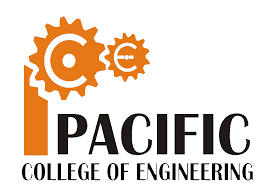
**Rajasthan Technical University, Kota**

*in partial fulfilment*

*for the award of the Degree of*

***Bachelor of Technology in***

***Computer Science & Engineering***

******

Submitted to: Submitted By: -

Dr. Prashant Sharma Prem Pratap Singh

16EPACS003

**Department of Computer Science Engineering**

**Pacific College of Engineering**

Udaipur, Rajasthan

Rajasthan Technical University

Kota, Rajasthan

***Candidate’s Declaration***

I hereby declare that the work, which is being presented in the Practical Training Seminar Report, entitled “**Computer Vision and Machine Learning”**

in partial fulfilment for the award of Degree of “*Bachelor of Technology”* in Computer Science Engineering and submitted to the **Department of Computer Science Engineering, Pacific College of Engineering, Udaipur, Rajasthan Technical University, Kota** is a record of my own training taken at **Adhoc Networks, Jaipur** carried under the guidance of Mr. Ashutoshh Singh, Sr. Data Science Specialist.

I have not submitted the matter presented in this report anywhere for the award of any other degree.

**Prem Pratap Singh**

Roll No.: - 16EPACS003

Enrolment No.: - 16E1PACSM30P003

Pacific College of Engineering, Udaipur

**ABSTRACT**

Adhoc Networks, a phenomenal place of interactive learning, strives for providing an outstanding platform for acquiring skills in the field of the latest and trending **open source technologies** in today’s digital advancing world. They are the leaders in **Data Science Domain Technologies** training, provide the best blend of delivery aspects, superb training content and best-of-breed services to professionals and interns for a bright career.

They are a group of experienced IT professionals providing the world class hands-on projects oriented corporate & technical training for organizations (Govt. and Public listed) clients & individuals in the domain of Data Science, Machine Learning, Big Data, AWS Cloud, Open Source Technologies & Red Hat Courses.

Here I learned about **Computer Vision** that deals with how computers can be made to gain high level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the **human visual system** can do. Computer vision tasks include method of acquiring, processing, analysing and understanding digital images, and extraction of high dimensional data from the real world in order to produce numerical and symbolic information.

I also learned how to train machine using machine learning models. **Machine Learning** is an application of **Artificial Intelligence** that provides system the ability to automatically learn and improve from experiences without being explicitly programmed. Machine learning focuses on the development of computer program that can access data and use it learn for themselves.

**ACKNOWLEDGMENET**

I take this opportunity to express my gratitude to all those people who have been directly and indirectly with me during the completion of this Practical Training Seminar.

I pay thank to Mr. Ashutoshh Singh who has given guidance and a light to me during this major project/seminar. His versatile knowledge about “Computer Vision & Machine Learning” has eased me in the critical times during the span of my major project/seminar/IT ACT Seminar.

I acknowledge here out debt to those who contributed significantly to one or more steps. I take full responsibility for any remaining sins of omission and commission.

Prem Pratap Singh

16EPACS003

B. Tech IV Year

Computer Science &

Engineering

**FIGURE LIST**

|  |  |  |
| --- | --- | --- |
|  | **Figure** | **Page No** |
| Figure 1.1 | SOURCE & DESTINATION IMAGES | 05 |
| Figure 1.2 | LANDMARK POINTS IN A FACE | 08 |
| Figure 1.3 | LANDMARK POINTS ON IMAGES | 09 |
| Figure 1.3 | TRIANGULATION OF SOURCE IMAGE | 10 |
| Figure 1.4 | TRIANGULATION OF DESTINATION IMAGE | 11 |
| Figure 1.5 | EXTRACT & WARP TRIANGLES | 11 |
| Figure 1.6 | LINK WARP TRIANGLES TOGETHER | 12 |
| Figure 1.7 | REPLACEMENT OF FACE ON DESTINATION FACE | 12 |
| Figure 1.8 | SEAMLESS CLONING ON IMAGE | 13 |
| Figure 2.1 | RAW GUI | 19 |
| Figure 2.2 | CREATION OF ALL FRAMES | 21 |
| Figure 2.3 | OUTPUT GUI INCLUDING ALL BUTTONS | 22 |
| Figure 2.4 | CAPTURING USER IMAGE FOR DATASET | 23 |
| Figure 2.5 | RECOGNIZING USER FOR ATTENDANCE | 24 |
| Figure 2.6 | SELECT FILE FOR SHARING ON TELEGRAM | 24 |
| Figure 2.7 | MORE OPTIONS | 25 |
| Figure 2.8 | EXIT MESSAGE BOX | 25 |
| Figure 3.1 | LED & RESISTOR CONNECTION | 27 |
| Figure 3.2 | LED & RESISTOR CONNECTION WITH BOLT MODULE | 27 |
| Figure 3.3 | BREADBOARD | 28 |
| Figure 3.4 | MULTIPLE LEDs CONNECTION ON BREADBOARD | 28 |
| Figure 3.5 | CONNECTION OF BOLT MODULE & BREADBOARD | 28 |
| Figure 3.6 | GENERATION OF API | 29 |
| Figure 3.7 | GUI TO CONTROL LED | 31 |
| Figure 3.8 | CREATING APPLET USING IFTTT | 33 |
| Figure 3.9 | CHOOSING ACTION SERVICE FOR APPLET REQUEST | 33 |
| Figure 3.10 | MAKING WEB REQUEST USING WEBHOOKS | 34 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**CONTENTS**

8

8

7

7

6

5

4

2

1

1-8

**PAGE NO.**

|  |  |
| --- | --- |
| **Title** |  |
| **PROJECT 1 FACE SWAPPING USING DELAUNAY TRIANGULATION** |  |
| 1.1 TAKE TWO IMAGES |  |
| [1.2 FIND LANDMARK POINTS ON BOTH IMAGES.](#page12) |  |
| [1.3 TRIANGULATION OF SOURCE IMAGE.](#page13)  1.4 TRIANGULATION OF DESTINATION IMAGE  1.5 EXTRACT & WARP TRIANGLES  1.6 LINK WARPPED TRIANGLES TOGETHER  1.7 REPLACE FACE ON THE DESTINATION IMAGE  1.8 SEAMLESS CLONING  1.9 CONCLUSION |  |

18

11

10

9-18

|  |  |
| --- | --- |
| **PROJECT 2 FACE RECOGNITION ATTENDANCE SYSTEM** |  |
| 2.1 FACE DETECTION |  |
| 2.2 GUI USING PYTHON TKINTER |  |
| 2.2.1 RAW GUI  2.2.2 CREATE ALL FRAMES  2.2.3 CREATE MENU  2.2.4 ADD FUNCTIONS TO BUTTON |  |
| 2.3 CONCLUSION |  |

22

21

20

20

19

19

19-28

|  |  |
| --- | --- |
| **PROJECT 3 LED BRIGHTNESS CONTROL USING BOLT IoT** |  |
| 3.1 MOTIVATION |  |
| 3.2 WHAT IS VOICE CONTROLLED LIGHTING ? |  |
| 3.3 HARDWARE & SOFTWARE REQUIREMENTS |  |
| 3.4 BOLT IoT |  |
| 3.5 HARDWARE CONFIGURATION |  |
| 3.6 GENERATING API |  |
| 3.7 CODE |  |
| 3.8 VOICE CONTROLLED LED AUTOMATION |  |
| 3.8.1 CREATING CONTROL URL |  |
| 3.8.2 SETTING UP IFTTT ACCOUNT |  |
| 3.9 CONCLUSION |  |

28

24

22

**Project 1**

**Face Swapping using Delaunay Triangulation**

Face swap refers to an activity in which a person’s face is swapped with the face of another person or animal or with an inanimate object. Many applications have been developed to use this feature like **face swap, snapchat face swapping** filter etc.

In mathematics and computational geometry, a **Delaunay triangulation** (also known as **Delone triangulation)** for a given set **P** of discrete points in a plane is a triangulation DT(**P**) such that no point in **P** is inside the circumcircle of any triangle in DT(**P**). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid silver triangles. The triangulation is named after **Boris Delaunay** for his on this topic from 1934.

In the plane, the Delaunay triangulation maximizes the minimum angle. Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other. However, the Delaunay triangulation does not necessarily minimize the maximum angle. The Delaunay triangulation also does not necessarily triangulation also does not necessarily minimize the length of the edges.

The language used for is **python** and the required python libraries are **dlib, cv2, numpy.**

The steps for face swapping using Delaunay triangulation are: -

* 1. **Take two images**

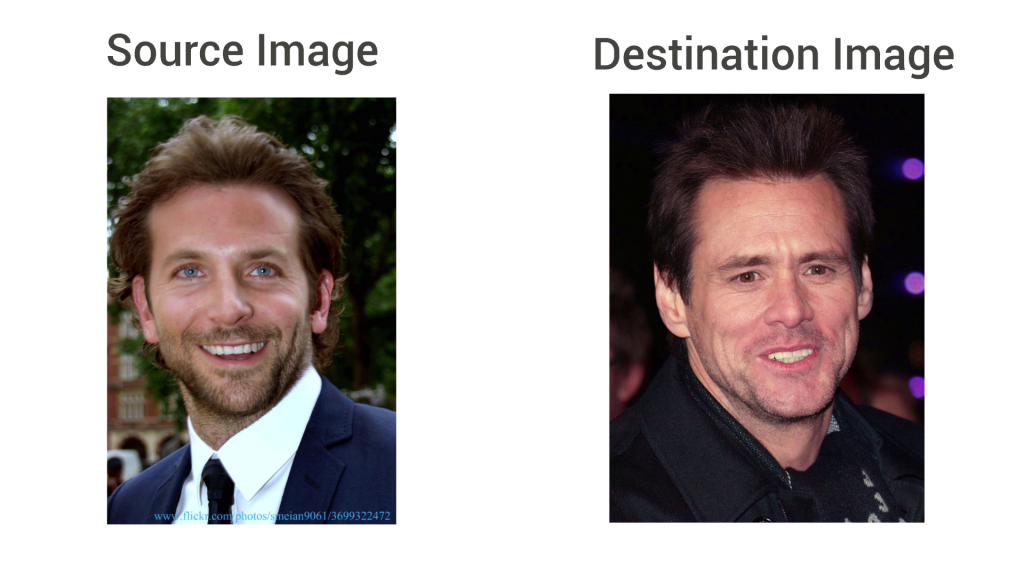
The “**source image**” is the one we take the face from and “**destination image**” is where we put the face extracted from the source image. These two images are read using the cv2 library.

Figure 1.1 Source & Destination Images

The code to read these images is

img = cv2.imread("bradley\_cooper.jpg")

img\_gray = cv2.cvtColor(img, cv2.**COLOR\_BGR2GRAY**)

img2 = cv2.imread("jim\_carrey.jpg")

img2\_gray = cv2.cvtColor(img2, cv2.**COLOR\_BGR2GRAY**)

* 1. **Find landmark points of both images**

Facial landmarks are used to localize and represent salient regions of the face, such as:

* Mouth
* Right eyebrow
* Left eyebrow
* Right eye
* Left eye
* Nose
* Jaw

Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. Detecting facial landmarks is a *subset* of the *shape prediction* problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape.

In the context of facial landmarks, the goal is to detect important facial structures on the face using shape prediction methods.

Detecting facial landmarks is therefore a two-step process:

* **Step #1:** Localize the face in the image.
* **Step #2:** Detect the key facial structures on the face ROI.

Step #1: **Face detection** can be achieved in a number of ways. We could use OpenCV’s built-in haar cascades. Or we might even use deep learning-based algorithms for face localization. In either case, the actual algorithm used to detect the face in the image doesn’t matter. Instead, what’s important is that through some method we obtain the face bounding box (i.e., the *(x, y)*-coordinates of the face in the image).

**Step #2: Detecting key facial structures in the face region.**

There are a variety of facial landmark detectors, but all methods essentially try to localize and label the facial regions

The facial landmark detector included in the dlib library is an implementation of the [*One Millisecond Face Alignment with an Ensemble of Regression Trees*](https://pdfs.semanticscholar.org/d78b/6a5b0dcaa81b1faea5fb0000045a62513567.pdf) paper by Kazemi and Sullivan (2014).

This method starts by using:

1. A training set of labelled facial landmarks on an image. These images are *manually labelled*, specifying **specific** *(x, y)*-coordinates of regions surrounding each facial structure.
2. *Priors*, of more specifically, the *probability on distance* between pairs of input pixels.

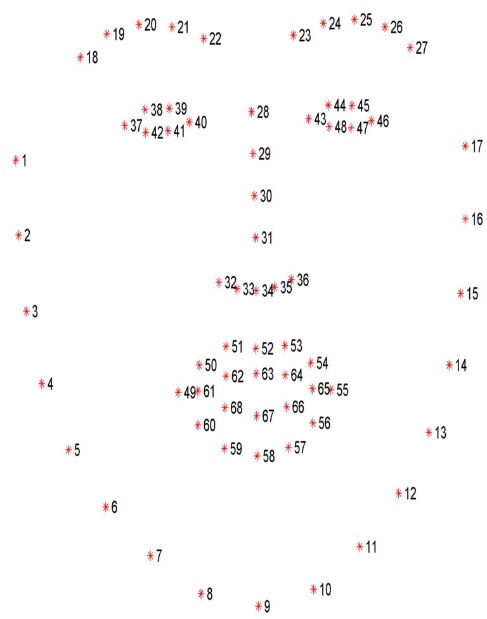
Given this training data, an ensemble of regression trees is trained to estimate the facial landmark positions directly from the *pixel intensities themselves* (i.e., no “feature extraction” is taking place).

The end result is a facial landmark detector that can be used to **detect facial landmarks in *real-time*** with **high quality predictions**.

**Understanding dlib’s facial landmark detector**

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of ***68 (x, y)-coordinates*** that map to facial structures on the face.

The indexes of the 68 coordinates can be visualized on the image below:

The code for facial landmark detection is:

*#load face detector and face landmark predictor*

detector = dlib.get\_frontal\_face\_detector ()

predictor= dlib.shape\_predictor(“face\_landmark\_68.dat”)

*#face1*

faces = detector (img\_gray)

for face in faces:

landmarks=predictor (img\_gray, face)

for n in range (0,68):

x = landmarks.part(n).x

y = landmarks.part(n).y

landmarks\_points.append ((x,y))

Figure 1.2 Landmark Points in a face



Figure 1.3 Landmark points on images

* 1. **Triangulation of source image**

Segment the face in the source image into triangles. The step is core of face swapping as to obtain final result simply exchange each triangle with the correspondent triangle of the destination image.

**What is the need to divide the face into triangles?**

It’s not appropriate just cut out the face from the source image and put it into the destination image as they have different size and perspective. Also it’s not correct to change the size and perspective right away because the face would lose the original proportions. Therefore, split the face into triangles, so that simply swapping of each triangle and in this way it will keep the proportions and also the it will match the expressions of the new face, like for example smile, close eyes or open mouth.

The triangulation performed is Delaunay triangulation which done using opencv and python.

*#delaunay triangulation*

*points = np.array(landmarks\_points, np.int32)*

*convexhull = cv2.convexHull(points)*

*rect = cv2.boundingRect(convexhull)*

*subdiv = cv2.Subdiv2D(rect)*

*subdiv.insert(landmarks\_points)*

*triangles = subdiv.getTriangleList()*

*triangles=np.array(triangles, dtype=np.int32)*

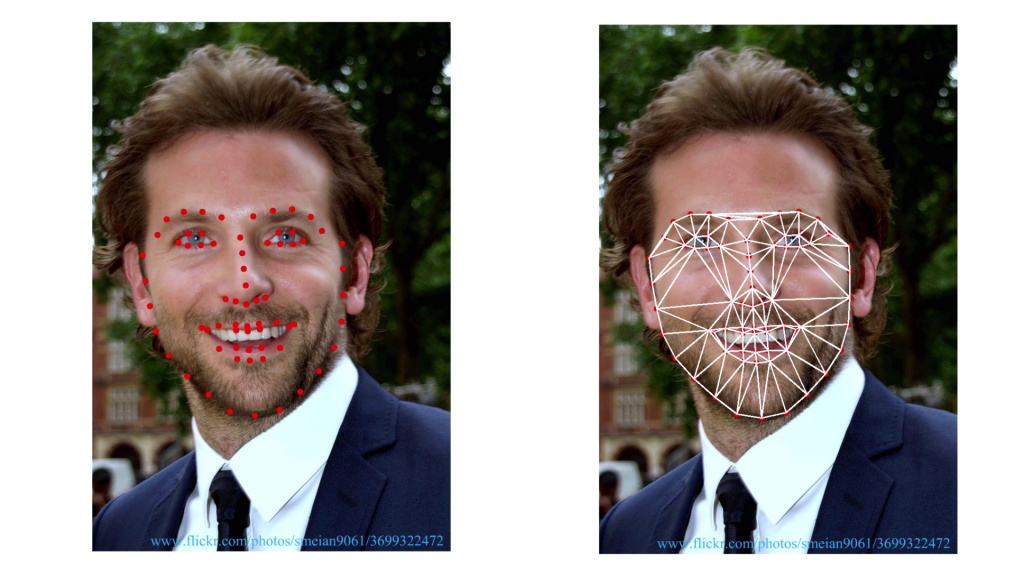


Figure 1.4 Triangulation of source image

* 1. **Triangulation of destination image**

The triangulation of destination image needs to have the same patterns of the triangulation of the source image. That means that the connection of the points has to be the same. So, after the triangulation of the source image, from that triangulation obtain the indexes of the landmark points and replicate the same triangulation on the destination image.

*#get the landmark point index of each triangle*

*indexes\_triangle = []*

*for t in triangles:*

*pt1 = (t[0],t[1])*

*pt2 = (t[2],t[3])*

*pt3 = (t[4],t[5])*

*index\_pt1 = np.where(points == pt1).all(axis=1)*

*index\_pt1 = extract\_index\_nparray(index\_pt1)*

*index\_pt2 = np.where(points == pt2).all(axis=1)*

*index\_pt2 = extract\_index\_nparray(index\_pt2)*

*index\_pt3 = np.where(points == pt3).all(axis=1)*

*index\_pt3 = extract\_index\_nparray(index\_pt3)*

*if index\_pt1 is not None and index\_pt2 is not None and index\_pt3 is not None:*

*triangle= [index\_pt1, index\_pt2, index\_pt3]*

*indexes\_triangles.append(triangle)*

Now loop through the triangles indexes and triangulate the destination face.

*for triangle\_index in indexes\_triangle:*

*tr1\_pt1 = landmarks\_points2(triangle\_index[0]]*

*tr1\_pt2 = landmarks\_points2(triangle\_index[1]]*

*tr1\_pt3 = landmarks\_points2(triangle\_index[2]]*

*triangle = np.array([tr1\_pt1, tr1\_pt2, tr1\_pt3], np.int32)*

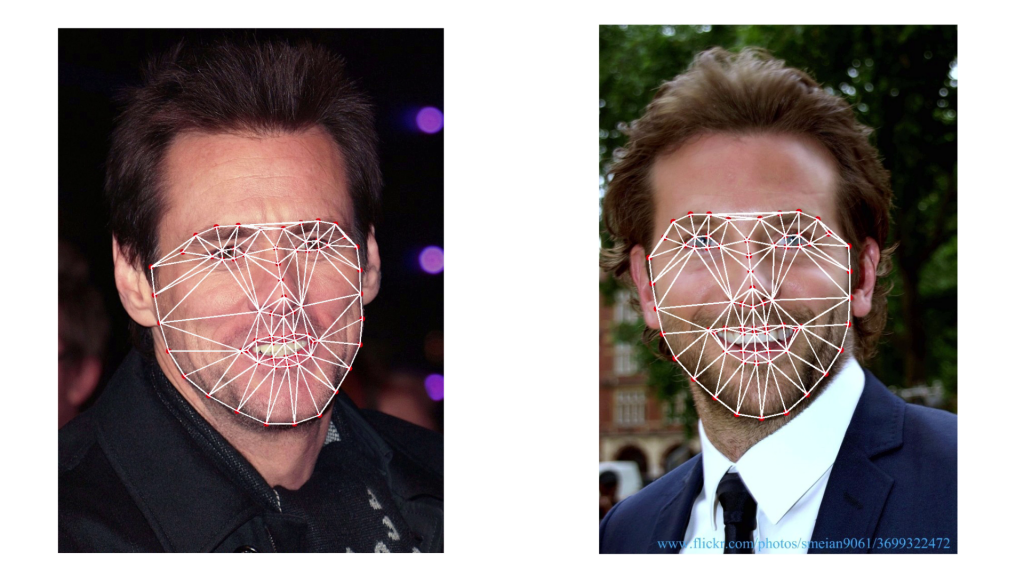


Figure 1.5 Triangulation of destination image

* 1. **Extract and warp triangles**

After the triangulation of both the faces, take the triangles of the source face and extract them. Also it is needed to take the coordinates of the triangles of destination face, so that the triangles of source face will have same size and perspective of the matching triangle on the destination face for warpping.

The code shows how to warp triangles of the source image

*# Warp triangles*

*points = np.float32(points)*

*points2 = np.float32(points2)*

***M*** *= cv2.getAffineTransform(points, points2)*

*warped\_triangle = cv2.warpAffine(cropped\_triangle,* ***M****, (w, h))*

*warped\_triangle = cv2.bitwise\_and(warped\_triangle, warped-triangle, mask =cropped\_tr2\_mask)*

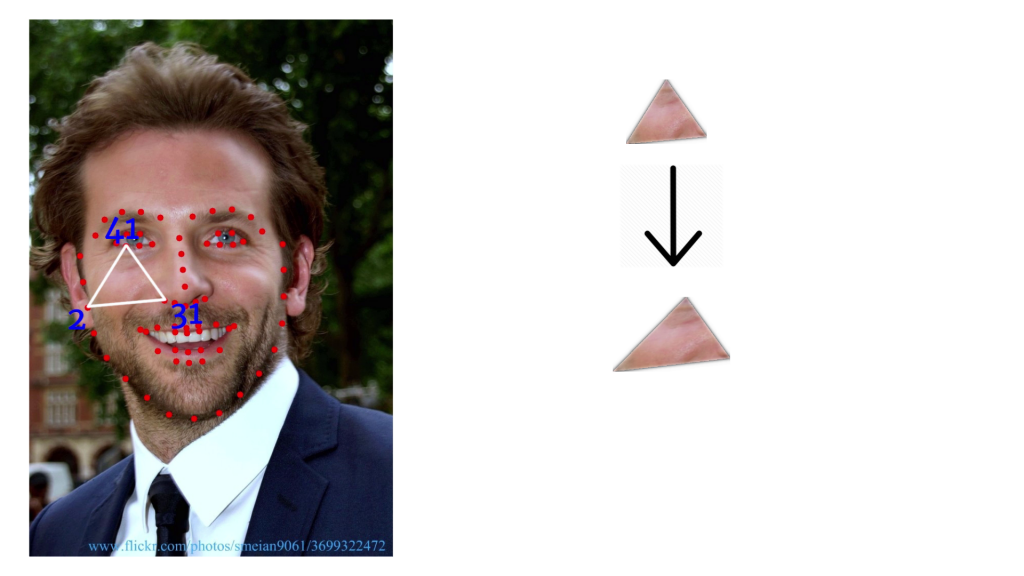
**

Figure 1.6 Extract and warp triangles

* 1. **Link the warpped triangles together**

After cutting and warping all the triangles, link them together. Now simply rebuild the face using the triangulation pattern, with the only difference that this time put the warped triangles. At the end of this the face is ready to be replaced.

*# Reconstructing destination face*

*img2\_new\_face = np.zeros((1155, 849, 3), np.uint8)*

*img2\_new\_face\_rect\_area = img2\_new\_face[y: y + h, x: x + w]*

*img2\_new\_face\_rect\_area\_gray = cv2.cvtColor(img2\_new\_face\_rect\_area, cv2.****COLOR\_BGR2GRAY****)*

*# Let's create a mask to remove the lines between the triangles*

*\_, mask\_triangles\_designed = cv2.threshold(img2\_new\_face\_rect\_area\_gray, 1, 255, cv2.****THRESH\_BINARY\_INV****)*

*warped\_triangle = cv2.bitwise\_and(warped\_triangle, warped\_triangle, mask=mask\_triangles\_designed)*

*img2\_new\_face\_rect\_area = cv2.add(img2\_new\_face\_rect\_area, warped\_triangle)*

*img2\_new\_face[y : y+h,x : x+w ] = img2\_new\_face\_rect\_area*

* 1. ****Replace the face on the destination image**

The face is now ready to be replaced and cut out the face of the destination image to make space for the new face. So, take the new face and the destination image without face and link them together.

Figure 1.6 Link triangles together

*# Face swapped (putting 1st face into 2nd face)*

*img2\_face\_mask = np.zeros\_like(img2\_gray)*

*img2\_head\_mask = cv2.fillConvexPoly(img2\_face\_mask, convexhull2, 255)*

*img2\_face\_mask = cv2.bitwise\_not(img2\_head\_mask)*

*img2\_head\_noface = cv2.bitwise\_and(img2, img2, mask=img2\_face\_mask)*

*result = cv2.add(img2\_head\_noface, img2\_new\_face)*

**

Figure 1.7 Replacing face on destination face

* 1. **Seamless cloning**

Finally the faces are correctly swapped and **it’s time to adjust the colors so that the source image fits the destination image.**

In opencv, there is a built in function called “**seamlessClone**” that does the operation automatically.

*(x, y, w, h) = cv2.boundingRect(convexhull2)*

*center\_face2 = (int((x + x + w) / 2), int((y + y + h) / 2))*

*seamlessclone = cv2.seamlessClone(result, img2, img2\_head\_mask, center\_face2, cv2.MIXED\_CLONE)*



Figure 1.8 Seamless Cloning

**1.9 Conclusion**

Face swapping using the method of Delaunay triangulation is successfully implemented for faces in sample images. Meanwhile different blending methods are experimented for a better swapping result.

**Project 2**

**Face Recognition Attendance System**

Being watched and/or monitored is a legitimate fear for many people. This is why facial recognition is slow to be accepted and welcomed in many industries. One industry or field of work that would benefit the most out of facial recognition, is that of attendance systems. The applications of Face Recognition Based Attendance System could be found in schools, offices, colleges, events and many other.

Most of us remember attendance at schools as the tutor calling out our names/registered numbers and us yelling back “PRESENT, MA’AM!”. All of that could soon change with the introduction of facial recognition in these systems.

## **How Face Recognition Based Attendance System works and advantages of Face Recognition Based Attendance System**

 Facial recognition systems need a database or a pre-recorded data set to compare captured images and identify faces. A recent report out of China stated that a school has installed facial recognition technology in its classrooms to monitor students’ behaviour.

In the same way, these cameras could be used to identify students and record their attendance in a class. With photos submitted to schools for various forms and documents, the students’ faces could be added to data sets to record their attendance in the respective classes.

Most offices and government buildings use biometric systems for logging in work hours and restricting access to only those people that are permitted. Although facial recognition is now being implemented in many countries for this purpose, they still largely rely on fingerprint sensors because of the lower chance of duplication and fraud. The downside to this is that recording a fingerprint is a task that requires relatively more effort.

By far the most interesting implementation of facial recognition in attendance systems is that of event attendance. The deployment of this system poses many benefits, especially for large events. Currently most events such as large corporate conferences and music concerts still largely rely on tickets, passes, and scanning of barcodes.

When audience numbers hit the late hundreds and even the thousands, the inefficiency of current systems begin to show. Checking tickets and scanning barcodes is a process that can take time depending on numerous variables.

Introducing facial recognition would cut down these inefficiencies since most modern facial recognition systems can perform the task in under two seconds. It is just a matter of collecting images of attendees at the time of issuing event passes.

This would also reduce the instances of ticket reselling as well as stolen or lost tickets being used by individuals that are not the official purchasers of said tickets.

**DIFFERENT APPROACHES OF FACE RECOGNITION**

There are two predominant approaches to the face recognition problem: Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature. Recognition algorithms can be divided into two main approaches:

1. Geometric: Is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features.
2. Photometric stereo: Used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface normals.

**2.1 Face Detection**

Face detection involves separating image windows into two classes; one containing faces (tarning the background). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin colour and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

The face detection system can be divided into the following steps: -

1. **Pre-Processing**: To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.
2. **Classification**: Neural networks are implemented to classify the images as faces or no faces by training on these examples. We use both our implementation of the neural network and the Matlab neural network toolbox for this task. Different network configurations are experimented with to optimize the results.
3. **Localization**: The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on: - Position Scale Orientation Illumination

Face recognition first involves face detection in the frame then recognizing that face using geometric or photometric stereo.

**2.2** **GUI using Python-Tkinter**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. There are several other popular Python GUI toolkits. Most popular are wxPython, PyQt, and PyGTK.

Install tkinter on Ubuntu using

**$ sudo apt-get install python3-tk**

The required python libraries to create complete attendance system are: -

* tkinter
* os
* cv2
* pickle
* face\_recognition
* datetime & time
* glob
* PIL.Image & PIL.ImageTk
* multiprocessing
* webbrowser
* telegram

**2.2.1 RAW GUI**

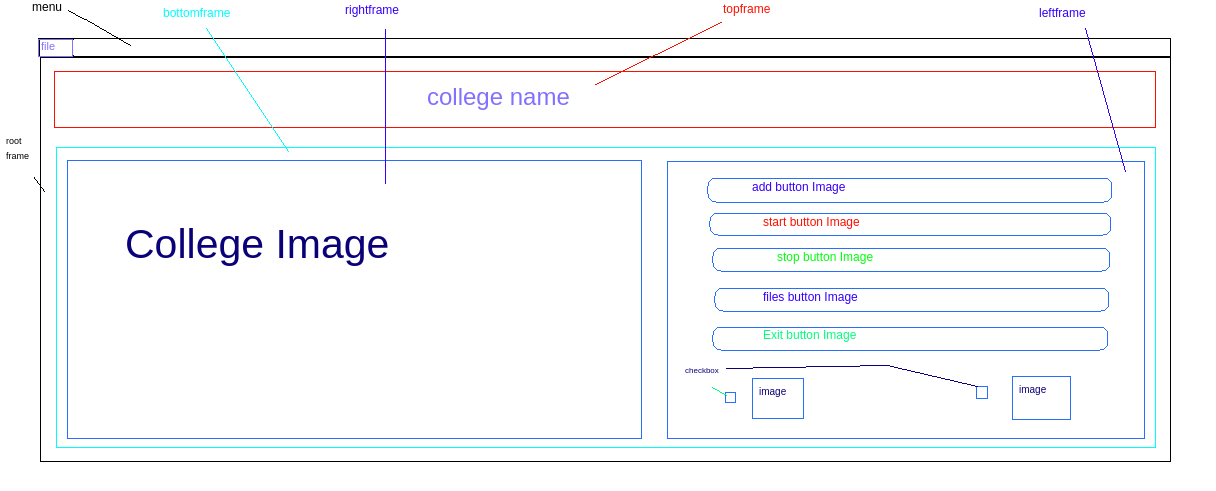
****

Figure 2.1 Raw GUI

**2.2.2 Create all frames**

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

**Syntax**

The simple syntax to create this widget −

w = Frame ( master, option, ... )

**Parameters**

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

The code to create all frames is

#import tkintter

from tkinter import \*

#Create window

root = Tk()

#window title

root.title("Pacific College of Engineering Attendance System")

#Create topframe with top side

topframe=Frame(root)

topframe.pack(side=TOP)

#Create bottomframe with bottom side

bottomframe=Frame(root)

#Create leftframe in bottomframe with left side

leftframe=Frame(bottomframe,bg='black')

leftframe.pack(side=LEFT)

#Create rightframe in bottomframe with right side

rightframe=Frame(bottomframe,padx=50)

rightframe.pack(side=RIGHT)

#Set window geometry width 1210 and height 750 and open position on screen left to 150 and top to 150

root.geometry("1210x750+150+150")

#stable main window on infinity time

root.mainloop()

Code for top frame is

#Create topframe on root window

topframe=Frame(root)

#create coustom font

my\_font=Font(family="Rekha",size=30,weight="bold",slant="italic")

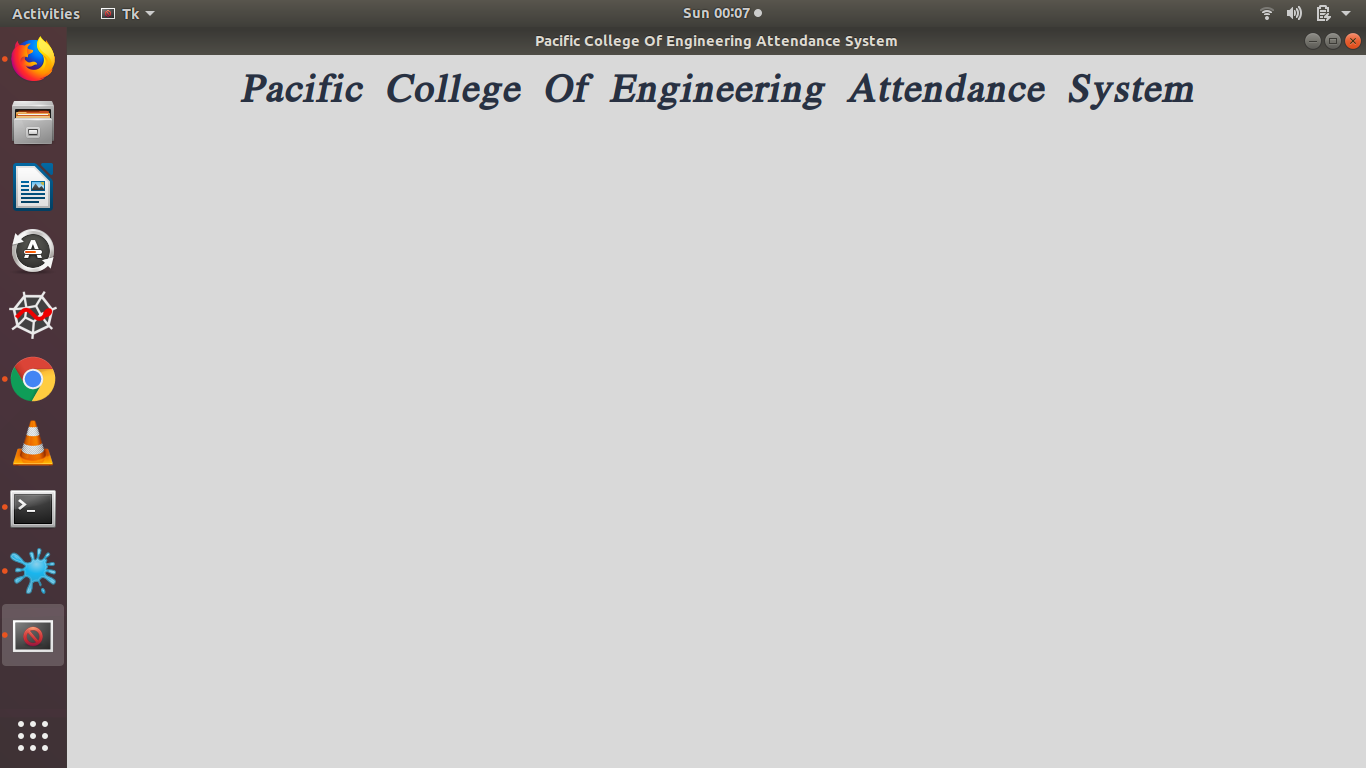
#create label on root window

label=Label(topframe,text="Pacific College Of Engineering Attendance System",font=my\_font,foreground="#283142").pack()

#pack topframe on top side

topframe.pack(side=TOP)

Figure 2.2 Creation of all frames



Code for leftframe in bottomframe which shows college logo in the output is

#create leftframe in bottomframe with left side

leftframe=Frame(bottomframe,bg='black')

#create a canvas for image on leftframe

canvas=Canvas(leftframe,width=627,height=620)

canvas.pack()

#photo for canvas with photo path

photo=PhotoImage(file=os.getcwd()+'/profile.png')

#create position of canvas and image start at NW

canvas.create\_image(0,0,image=photo,anchor=NW)

leftframe.pack(side=LEFT)

Code for rightframe in bottomframe which have the buttons for different functionalities like **add user, start attendance, share files, view files** etc. Check buttons are also created to set camera by entering the IP of camera, to upload photo of a person, to set time duration of taking attendance.

#Create rightframe in bottomframe with right side

rightframe=Frame(bottomframe,padx=50)

#create all image path for buttons and checkbox images

photo1=PhotoImage(file=os.getcwd()+'/useradd.png')

photo2=PhotoImage(file=os.getcwd()+'/start.png')

photo3=PhotoImage(file=os.getcwd()+'/telegram1.png')

photo4=PhotoImage(file=os.getcwd()+'/folder.png')

photo5=PhotoImage(file=os.getcwd()+'/exit.png')

photo6=PhotoImage(file=os.getcwd()+'/clock.png')

photo7=PhotoImage(file=os.getcwd()+'/upload.png')

photo8=PhotoImage(file=os.getcwd()+'/cam.png')

#button bt1 for add new entry

Bt1=Button(rightframe,text="New Entry",command=Open\_new,image=photo1,activebackground="navy",fg='grey', bd=8,width=10)

Bt1.pack(fill=X)

#button bt2 for start webcam

Bt2=Button(rightframe,text="Start",image=photo2,command=start,activebackground="navy",fg='grey', bd=8)

Bt2.pack(fill=X)

#button bt3 for sharing file on telegram

Bt3=Button(rightframe,text="Share file on telegram",image=photo3,command=share,activebackground="navy",fg='grey', bd=8)

Bt3.pack(fill=BOTH,pady=10)

#button bt4 for seen files

Bt4=Button(rightframe,text="files",width=10,image=photo4,command=open\_file,activebackground="navy",fg='grey', bd=8)

Bt4.pack(fill=X,pady=10)

#button bt5 for Exit programme

Bt5=Button(rightframe,text="EXIT",image=photo5,command=exit,activebackground="navy",fg='grey', bd=8)

Bt5.pack(fill=X,pady=10)

#check\_bt for set time period in minutes

h=StringVar()

check\_bt=Checkbutton(rightframe,text="Set Time",variable=h, offvalue="uncheck",onvalue="check",activeforeground="green",width=120,image=photo6,compound=TOP)

check\_bt.pack(side=LEFT)

#check\_bt2 for upload images if person(student notpresent ) or offline making data

M=StringVar()

check\_bt2=Checkbutton(rightframe,text="Upload Image",variable=M, offvalue="uncheck",onvalue="check",activeforeground="green",width=120,image=photo7,compound=TOP)

check\_bt2.pack(side=LEFT)

#checkbutton t set camera

k=StringVar()

check\_bt=Checkbutton(rightframe,text="Set Camera",variable=k, offvalue="cameraoff",onvalue="cameraon",activeforeground="green",width=130,image=photo8,compound=TOP)

check\_bt.pack(side=RIGHT)

rightframe.pack(side=RIGHT)

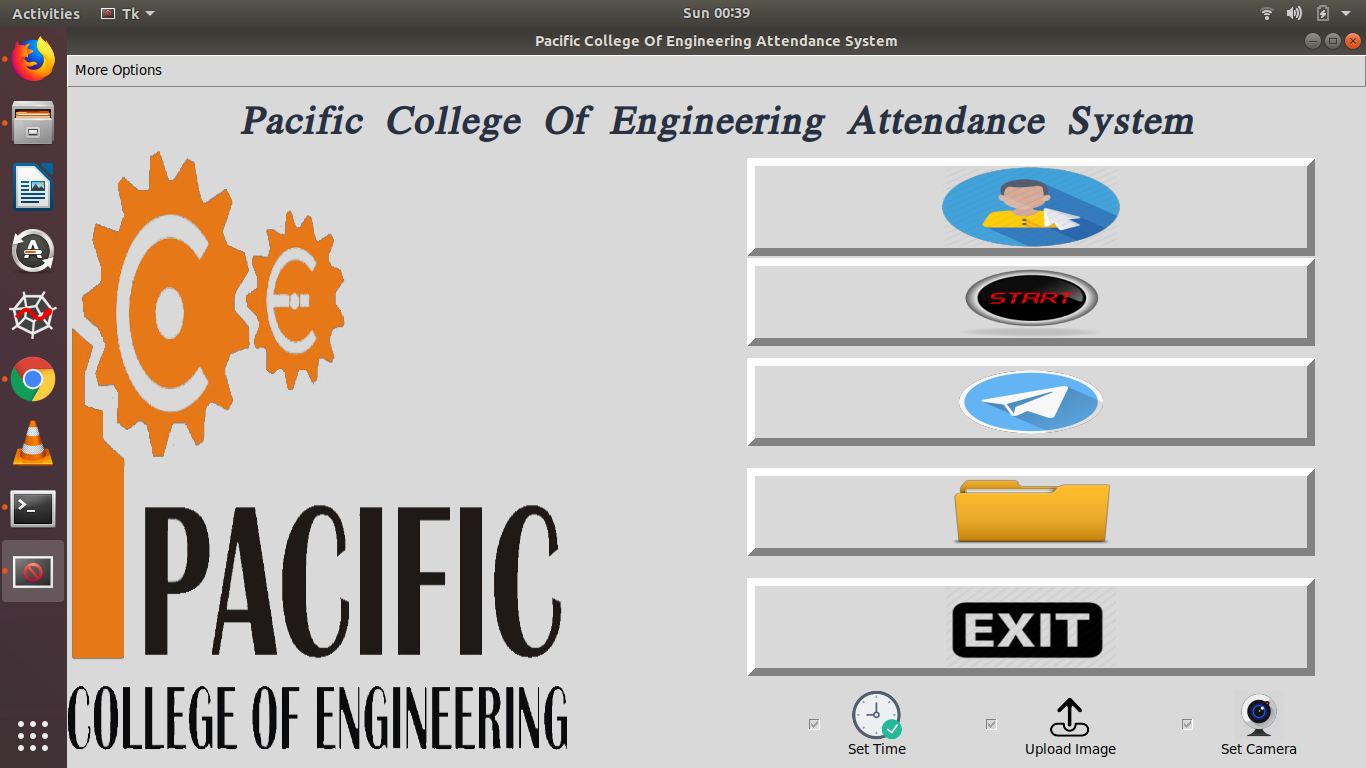


Figure 2.3 Output GUI including all buttons

**2.2.3 Create Menu**

Create menu on root having multiple options like **today’s attendance count, attendance count of student, display images** etc.

#create a main\_menu on root

main\_menu=Menu(root)

#config main\_menu with root

root.config(menu=main\_menu)

#Create fileMenu on main\_menu and tearoff = False means not show deses point on menu

fileMenu=Menu(main\_menu,tearoff=False)

main\_menu.add\_cascade(label="More Options",menu=fileMenu)

fileMenu.add\_command(label="Open files",command=open\_file)

fileMenu.add\_separator()

fileMenu.add\_command(label="Images",command=Images\_show)

fileMenu.add\_separator()

fileMenu.add\_command(label="Today's Attendance",command=today\_attendance)

fileMenu.add\_command(label="Attendance Count",command=Attendance\_count)

fileMenu.add\_separator()

Newfile=Menu(fileMenu,tearoff=False)

Newfile.add\_command(label="College Website",command=open\_site)

Newfile.add\_command(label="About Me",command=openweb)

fileMenu.add\_cascade(label="More options",menu=Newfile)

**2.2.4 Add functions to button**

Functions are defined for different buttons created to perform their functionality.

* **Add User** button takes input student name and student id, then starts camera to capture image of the user and saves it in the folder created using student name and id. If the check button is checked then on clicking add user button, a dialogbox will open to select the image want to add to dataset.

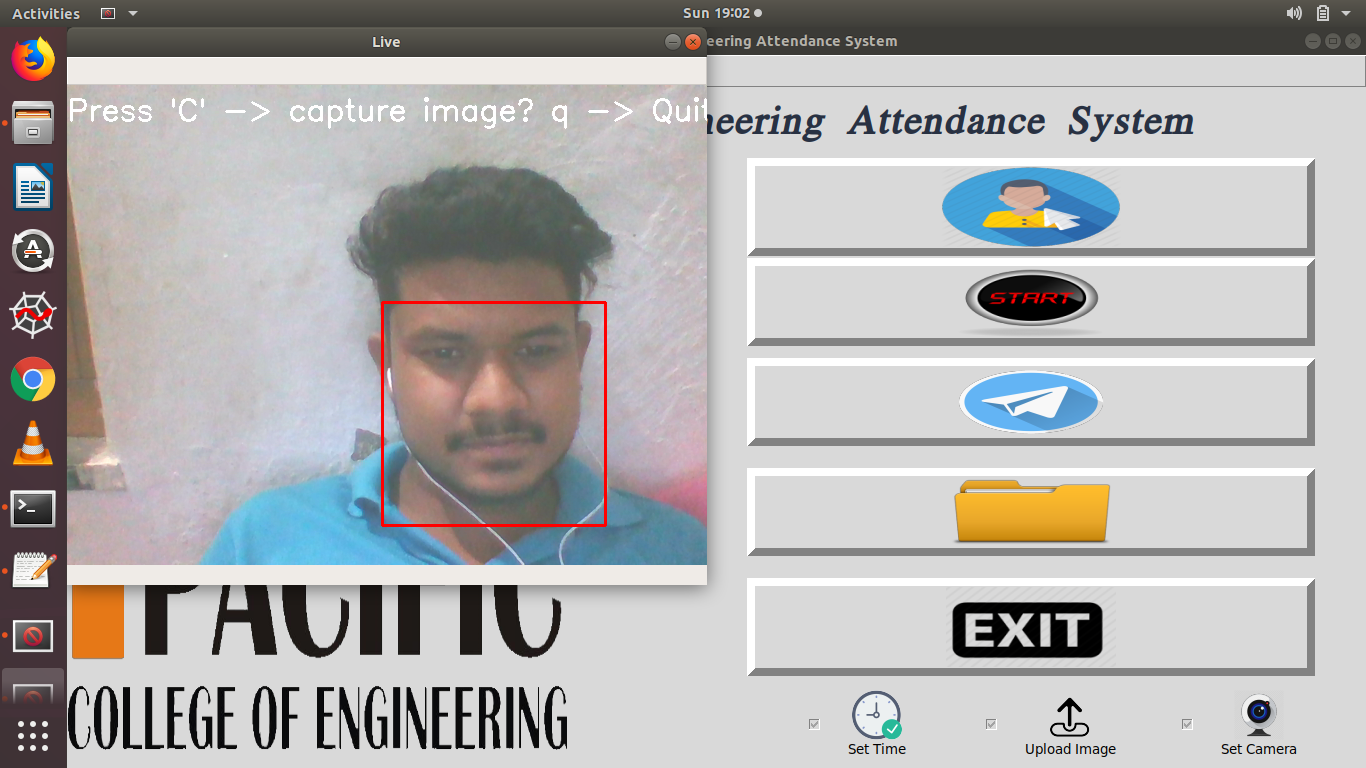


Figure 2.4 Capturing user image for dataset

* **Start** button is the main button to take attendance. Camera will open and if the student in front of camera matches with one of the student’s facial data from the dataset then, his attendance is marked along with time. A file is created with date as the name having complete attendance of the day.

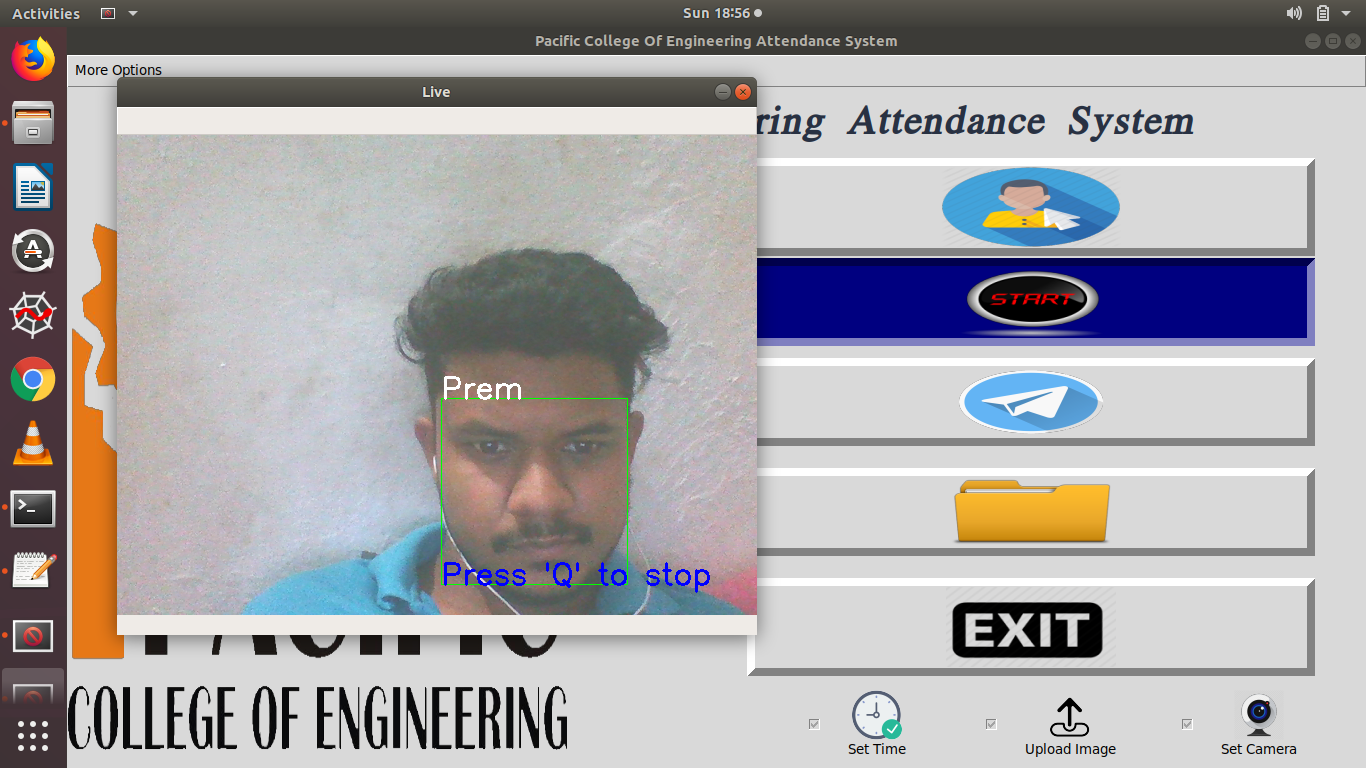


Figure 2.5 Recognizing user for attendance

* **Telegram share** button is the through which files can be shared in the group by the chat bot created on telegram. Every member of the group can access those files and check attendance of the students.

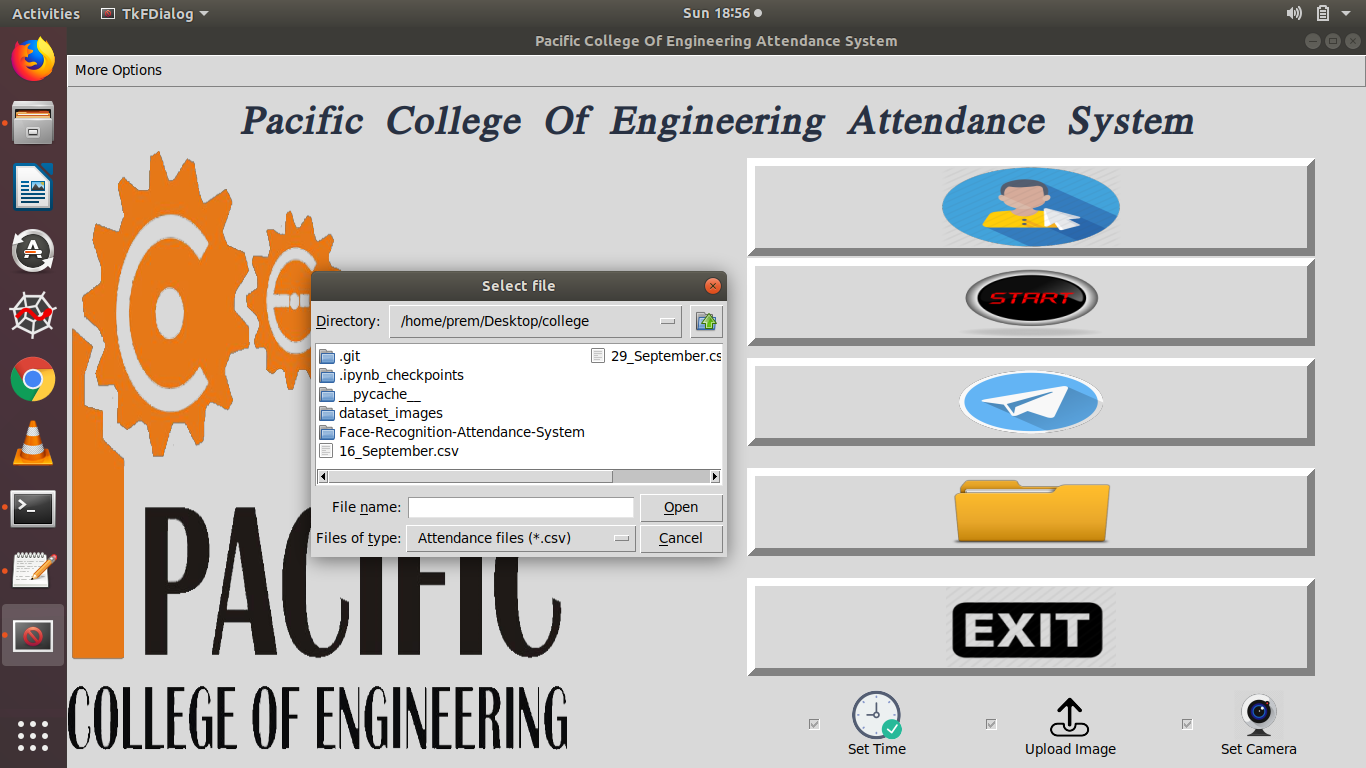
****

Figure 2.6 Select file for sharing on telegram

* **Folder** icon shows the button to access all files created for reading.
* More options include options to open files and image, find the total attendance of the student till that period, and also to direct access the today’s attendance file. There are options to visit the college website also.

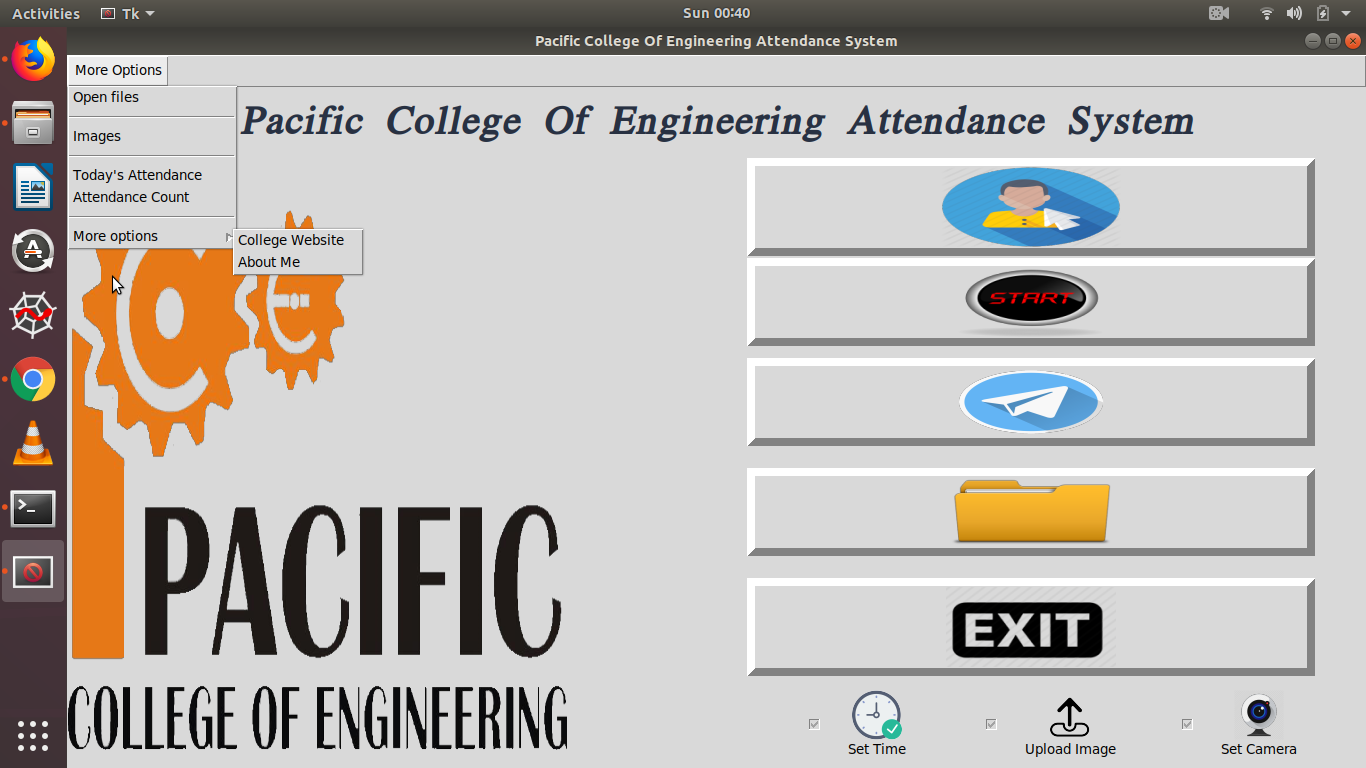


Figure 2.7 More Options

* **Exit** button closes the complete attendance system. A message box appeared asking to exit the system on clicking exit button.

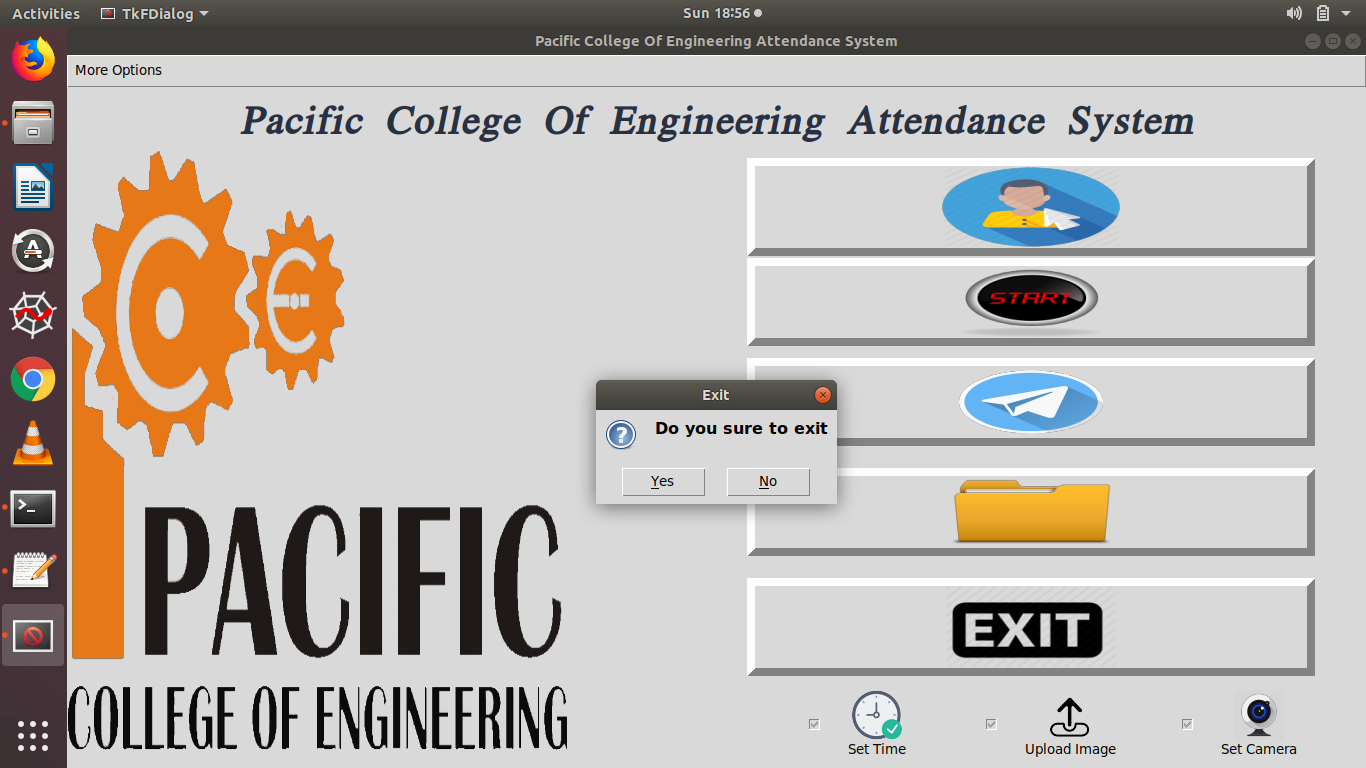


Figure 2.8 Exit message box

**2.3 Conclusion**

Summing it up, every new facial recognition technology represents huge perspectives and promises for the future evolution. It’s highly possible that in a couple of years such systems would be able to process gestures, expressions, gait patterns, palm & ear prints, voice and scent signatures. Not to mention that humans can’t do what a computer is capable off – like comparing simultaneously pictures of many individuals against a database of thousands.

**Project 3**

**LED Brightness Control Using Bolt IoT**

The **Internet of Things** (**IoT**) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The definition of the Internet of things has evolved due to the convergence of the multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation) and others all contribute to enabling the Internet of Things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "**smart home**", covering devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

 IoT describes a world where just about anything can be connected and communicate in an intelligent fashion. In other words, with the internet of things, the physical world is becoming one big information system.

**3.1 Motivation**

In past days people use different kinds of lights for e.g. bright light while they work and low light while they sleep or watch movies. Controlling brightness of light using IoT is an application of smart home.

IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems.Long-term benefits could include energy savings by automatically ensuring lights and electronics are turned off.

A smart home or automated home could be based on a platform or hubs that control smart devices and appliances.

**3.2 What is voice controlled lighting ?**

A virtual digital assistant is software that can perform tasks for consumers such as looking up information on the internet, playing media and buying products online. It uses speech recognition technology to respond to voice commands.

Amazon Alexa, Apple’s Siri, Google Assistant and Microsoft’s Cortana are the most popular virtual assistant platforms. Market research firm Tractica estimated 390 million people used virtual assistants in 2015 and projected this number to grow to 1 billion in 2018 and 1.8 billion in 2021.

Until now, smart lighting in the home was controlled using apps and devices such as keypads, dimmers, switches and sensors. With these methods, homeowners can operate and schedule their lighting. By using a system with a compatible virtual assistant app/device, users can now also control their lighting, shades, thermostats, audio/video, and other smart devices using voice commands. The benefit, of course, is convenience and enhanced lifestyle.

For voice-controlled lighting, what’s needed is a lighting or home control system, compatible virtual assistant device (phone or speaker) and app, and a robust Wi-Fi connection.

User voice commands pass through the virtual assistant’s external cloud-based service to the control provider’s cloud-based service, which interoperate using an application programming interface. The control provider then sends the appropriate control signal to the controller in the home to execute the command. Control response should be almost immediate.

**3.3 Hardware & software requirements**

* 5mm LED
* Bolt IoT Wifi Module
* USB-A to Micro- USB cable
* Breadboard
* Jumper wires
* Resistor 10k ohm
* IFTTT software app
* Google assistant SDK

**3.4 Bolt IoT**

Bolt is an IoT platform to easily and quickly build products and services. Bolt comes with a WiFi/GSM chip and a cloud platform which helps user connect devices and sensors to the Internet. User can configure such a system over the Bolt Cloud to receive, store and visualise the data over graphs. User can also connect actuators such as motors, light bulb and control them over the Internet. They also offer APIs to fetch data and control the devices from any platform. With Bolt Cloud user can control and monitor them over the internet, create personalised dashboards to visualise the data, monitor the device health, run machine learning algorithms and lot more. Build scalable IoT systems in just a few days time.

**BOLT functionalities**

* Station mode in which it can connect to WiFi networks.
* When not connected to any WiFi network it hosts its own WiFi hotspot to which users can connect.
* Commands an ESP8266 modules to do all GPIO and UART based tasks which runs at a frequency of 80Mhz.

**3.5 Hardware Configuration**

A **light-emitting diode** (**LED**) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor.White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.

**Connecting a single LED**

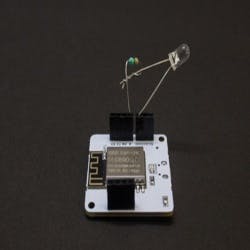
Take one LED and a 10K resistor. Connect one leg of resistor to the positive leg (longer leg) of LED and then put the another leg of resistor into one of the GPIO pin. I am using pin 0. Now put the negative leg(shorter leg) into the GND(ground) of Bolt IoT WiFi module. **Always use resistor with LED otherwise your LED will be burnt if you try connect without resistor.**

Figure 3.2 LED & resistor connection with Bolt

module

Figure 3.1 LED & resistor connection

**Connecting multiple LEDs**

To connect multiple LED's, connections have to made on to on breadboard. The motive remains same to connect resistor with the positive leg of LED, in case of multiple LED’s pin configuration of the breadboard should be considered. The first and second last row in the breadboard denotes positive pins. Every positive supply must be connected to these pins. The second and the last row denotes negative pins. Every ground must be connected to these pins. The all positive pins from 0 to 30 are connected internally in horizontal manner and similarly negative pins are connected internally. The pins in rows: A, B, C, D, E are connected internally in vertical manner. Setup the Bolt WiFi module and connect it to Bolt cloud. The negative wire (black wire) is connected to pin GND of module and the positive wire (red wire) is connected to GPIO pin 0 as shown in the above pictures.

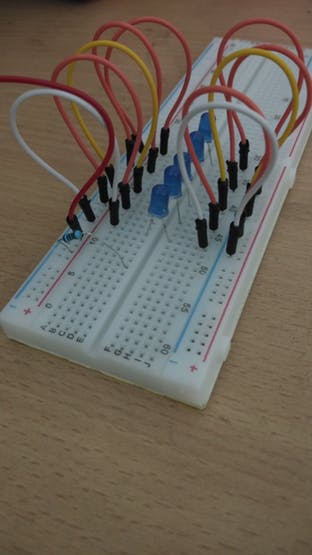
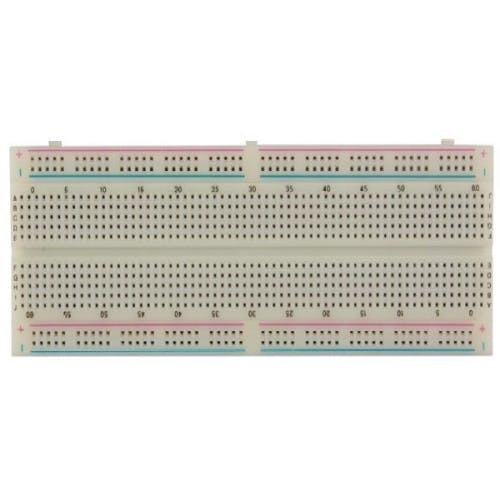
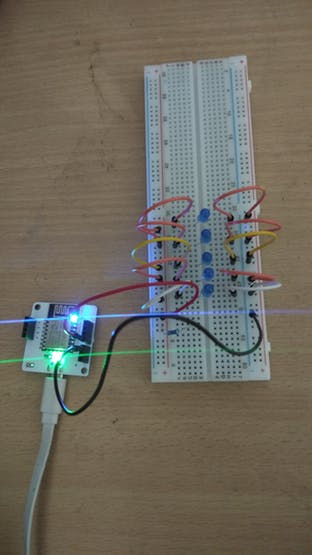


Figure 3.4 Multiple LEDs connection on breadboard

Figure 3.3 Breadboard

Figure 3.5 Connection of Bolt module & breadboard

**3.6 Generating API key**

Login to [https://cloud.boltiot.com.](https://cloud.boltiot.com/) And if user haven't generated API key click on generate API key and make sure to enable it then copy and paste it somewhere safely. API key may look like this: - 44b2de6b-7e68-40e7-a27f-814b58afe008.

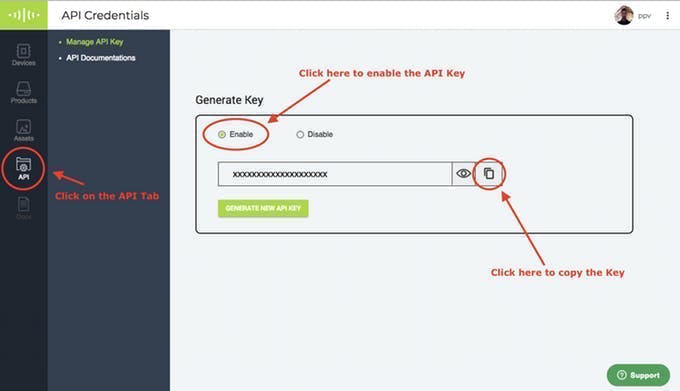


Figure 3.6 Generation of API

Note bolt ID which can be easily found on the dashboard & it may look like like this: - BOLT1234567

**3.7 Code**

Creating a GUI using python standard library tkinter is easy. To install tkinter on Ubuntu and run the command

**sudo apt install python3-tk**

The complete code is

#!/usr/bin/env python3

from tkinter import \*

from boltiot import Bolt

import json

import os

start=Tk()

start.geometry("500x350+100+100")

start.title("Led ControL along with brightness")

def on():

response=mybolt.isOnline()

data=json.loads(response)

if data["value"]=="online":

mybolt.digitalWrite('0','HIGH')#0 is the gpio pin

else:

print("Your device is offline")

def off():

response=mybolt.isOnline()

data=json.loads(response)

print(data)

if data["value"]=="online":

mybolt.digitalWrite('0','LOW')#0 is the gpio pin

else:

print("Your device is offline")#function to change intensity of led

def set\_value():

value=s.get()

response=mybolt.isOnline()

if 'online' in response:

mybolt.analogWrite('0',value)#first arguement is pin number and second is intensity value from 0 to 255

else:

print("Your device is offline")

#Creating a top frame

topframe=Frame(start,width=300,height=150)

lab=Label(topframe,text="Led Control")

lab.pack(fill=X)

photo1=PhotoImage(file=os.getcwd()+'/on1.png')

photo2=PhotoImage(file=os.getcwd()+'/off1.png')

Butt1=Button(topframe,text="ON",command=on,image=photo1,activebackground="navy",fg='grey', bd=8)

Butt1.pack()

Butt2=Button(topframe,text="OFF",command=off,image=photo2,activebackground="navy",fg='grey', bd=8)

Butt2.pack()

topframe.pack()

#creating bottom frame for led brightness control

bottomframe=Frame(start)

label=Label(bottomframe,text="Led Brightness Manager")

label.pack(fill=X)

s=Scale(bottomframe,from\_=0,to=255,length=200,width=20,sliderlength=20,orient=HORIZONTAL)

s.pack()

butt3=Button(bottomframe,text="Set Value",command=set\_value)

butt3.pack()

bottomframe.pack(side=BOTTOM)

#Bolt led control code

API="Your API key"

device\_id="Your device id"

mybolt=Bolt(API,device\_id)

start.mainloop()

The images used for buttons must be placed in the same directory in which code is placed. The output after running the code will look like this.



Figure 3.7 GUI to control LED

Control LED using this interface. Click onto “**TURN ON**” the LEDs and click on to “**TURN OFF**” the LED.

**3.8 Voice Controlled LED Automation**

To make voice control LED automation, a software IFTTT (**If** **T**his **T**hen **T**hat).

**3.8.1 Create control url**

Control url is created using API key and Bolt device id. Control URL to control LED using google assistant will look like this:[https://cloud.boltiot.com/remote/**YOUR\_API\_KEY/**digitalWrite?pin=**GPIO\_PIN\_YOU\_SELECTED**&state=**HIGH**&deviceName=**YOUR\_DEVICE\_ID**](https://cloud.boltiot.com/remote/YOUR_API_KEY/digitalWrite?pin=GPIO_PIN_YOU_SELECTED&state=HIGH&deviceName=YOUR_DEVICE_ID)

For e.g. [https://cloud.boltiot.com/remote/44b2de6b-7e68-40e7-a27f-814b58afe008/digitalWrite?pin=0&state=HIGH&deviceName=BOLT1234567](https://cloud.boltiot.com/remote/44b2de6b-7e68-40e7-a27f-814b58afe008/digitalWrite?pin=1&value=0&deviceName=BOLT1234567)

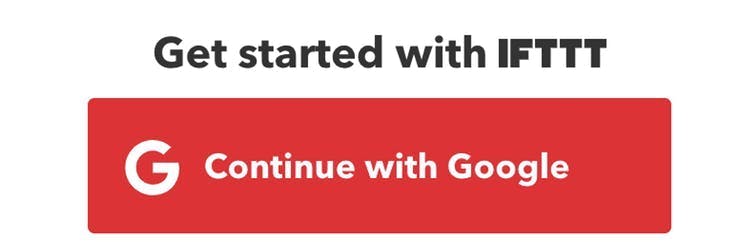
The above URL is to **turn on** the LED. To **turn off** the LED just replace the state from **HIGH** to **LOW.**

**3.8.2 Setting up IFTTT Account**

Go to ifttt.com by clicking this URL :- [https://ifttt.com.](https://ifttt.com/) Create account on it and then click on create new applet or use this URL :- <https://ifttt.com/create>

Login using Gmail account. Make sure to use same account which you'll be using on your mobile to interact with Google Assistant.

User can also install IFTTT android app which is much easier to use.



Click on '+This' to create the trigger.



Choose Google Assistant -> Say Specific Phrase. 

Type the phrase user want to trigger the action. Make sure to specify the trigger command in different ways for example.

1. Turn on the lights

2. Buddy turn the lights on

3. Dude get me in light.

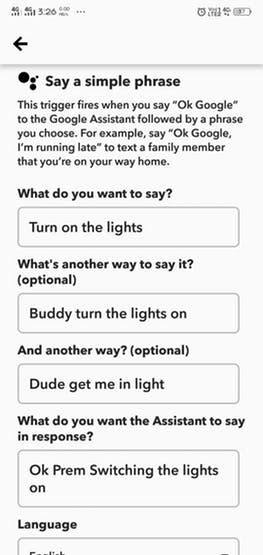


Figure 3.8 Creating applet using IFTTT

Now click on '+That'.



Select **Webhooks** and then Make a web request.

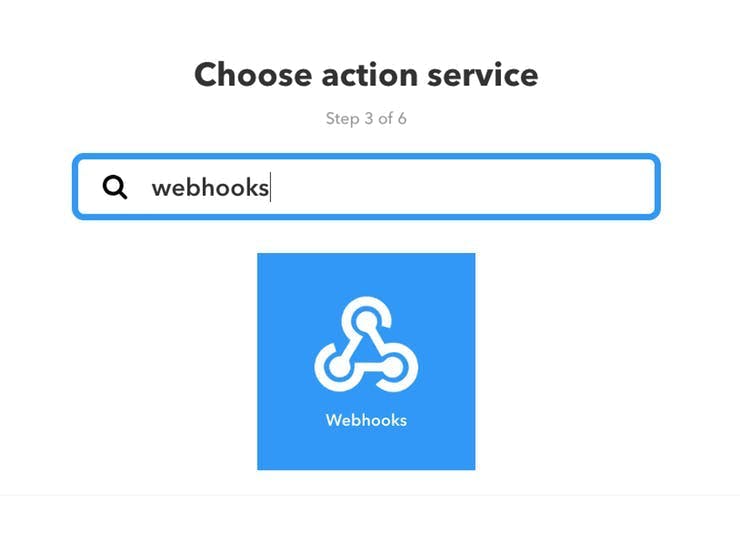


Figure 3.9 Choosing action service for applet request

Enter the control url created using API. Make sure to change the API Key and device name.

Method will be GET.

Content type will be Application/json.



Figure 3.10 Making web request using webhooks

Click on 'Create Action' and then Click on 'Finish'.

So virtual assistant (google assistant) is now setup to turn on the LED. Similarly create applet to turn off the LED using IFTTT app.

**3.9 Conclusion**

Virtual assistants and voice control use is expected to continue to grow, which will impact and add more value to home automation and IoT industry. It will enhance lifestyle of the people.