In [1]:
```
import  cv2
import numpy as np
from keras.models import load_model
from keras.applications import VGG16
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of i
ssubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).
type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
EMOTION_DICT = {1:"ANGRY", 2:"DISGUST", 3:"FEAR", 4:"HAPPY", 5:"NEUTRAL", 6:"SAD", 7:"SURPRISE"}
model_VGG = VGG16(weights='imagenet', include_top=False)
model_top = load_model("../Data/Model_Save/model.h5")
```

In [3]:
```
def return_prediction(path):
    #converting image to gray scale and save it
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cv2.imwrite(path, gray)

    #detect face in image, crop it then resize it then save it
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        face_clip = img[y:y+h, x:x+w]
        cv2.imwrite(path, cv2.resize(face_clip, (350, 350)))

    #read the processed image then make prediction and display the result
    read_image = cv2.imread(path)
    read_image = read_image.reshape(1, read_image.shape[0], read_image.shape[1], read_image.shape[2])
    read_image_final = read_image/255.0
    VGG_Pred = model_VGG.predict(read_image_final)
    VGG_Pred = VGG_Pred.reshape(1, VGG_Pred.shape[1]*VGG_Pred.shape[2]*VGG_Pred.shape[3])
    top_pred = model_top.predict(VGG_Pred)
    emotion_label = top_pred[0].argmax() + 1
    return EMOTION_DICT[emotion_label]
```

In [4]:
```
def rerun(text, cap):
    while(True):
        ret, img = cap.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, "Last Emotion was "+str(text), (95,30), font, 1.0, (255, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(img, "Press SPACE: FOR EMOTION", (5,470), font, 0.7, (255, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(img, "Hold Q: To Quit", (460,470), font, 0.7, (255, 0, 0), 2, cv2.LINE_AA)

        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for x,y,w,h in faces:
            cv2.rectangle(img, (x,y), (x+w, y+h), (255, 0, 0), 2)

        cv2.imshow("Image", img)

        if cv2.waitKey(1) == ord(' '):
            cv2.imwrite("test.jpg", img)
            text = return_prediction("test.jpg")
            first_run(text, cap)
            break

        if cv2.waitKey(1) == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
```

In [5]:
```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

def first_run(text, cap):
    while(True):
        ret, img = cap.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, "Last Emotion was "+str(text), (95,30), font, 1.0, (255, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(img, "Press SPACE: FOR EMOTION", (5,470), font, 0.7, (255, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(img, "Hold Q: To Quit", (460,470), font, 0.7, (255, 0, 0), 2, cv2.LINE_AA)

        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for x,y,w,h in faces:
            cv2.rectangle(img, (x,y), (x+w, y+h), (255, 0, 0), 2)

        cv2.imshow("Image", img)

        if cv2.waitKey(1) == ord(' '):
            cv2.imwrite("test.jpg", img)
            text = return_prediction("test.jpg")
            rerun(text, cap)
            break

        if cv2.waitKey(1) == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
```

In [6]:
```
first_run("None", cap)
```