In [1]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\pravallika\anaco
nda3\lib\site-packages (4.8.0.76)
Requirement already satisfied: numpy>=1.19.3 in c:\users\pravallika\anaco
nda3\lib\site-packages (from opencv-python) (1.22.4)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install deepface
```

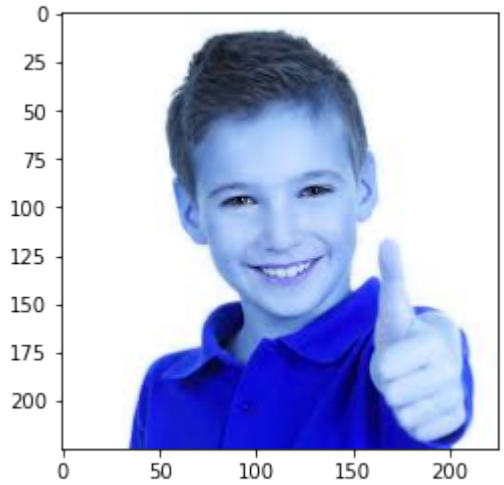
```
conda3\lib\site-packages (from deepface) (1.1.2)
Requirement already satisfied: gdown>=3.10.1 in c:\users\pravallika\an
aconda3\lib\site-packages (from deepface) (5.0.1)
Requirement already satisfied: keras>=2.2.0 in c:\users\pravallika\ana
conda3\lib\site-packages (from deepface) (2.13.1)
Requirement already satisfied: tqdm>=4.30.0 in c:\users\pravallika\ana
conda3\lib\site-packages (from deepface) (4.62.3)
Requirement already satisfied: opencv-python>=4.5.5.64 in c:\users\pra
vallika\anaconda3\lib\site-packages (from deepface) (4.8.0.76)
Requirement already satisfied: pandas>=0.23.4 in c:\users\pravallika\a
naconda3\lib\site-packages (from deepface) (1.3.4)
Requirement already satisfied: retina-face>=0.0.1 in c:\users\pravalli
ka\anaconda3\lib\site-packages (from deepface) (0.0.14)
Requirement already satisfied: tensorflow>=1.9.0 in c:\users\pravallik
a\anaconda3\lib\site-packages (from deepface) (2.13.0)
Requirement already satisfied: mtcnn>=0.1.0 in c:\users\pravallika\ana
conda3\lib\site-packages (from deepface) (0.1.1)
Requirement already satisfied: fire>=0.4.0 in c:\users\pravallika\anac
onda3\lib\site-packages (from deepface) (0.5.0)
Requirement already satisfied: numpy>=1.14.0 in c:\users\pravallika\an
```

In [3]:
```
from deepface import DeepFace
```

In [4]:
```
#implementation using python,opencv,deepface
import cv2
img=cv2.imread(r"C:\Users\Pravallika\OneDrive\Pictures\happyboy.jpg")
```

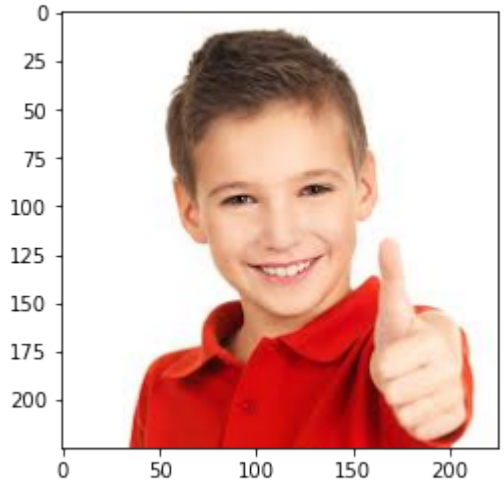
In [5]:
```
import matplotlib.pyplot as plt
plt.imshow(img)
```

Out[5]: <matplotlib.image.AxesImage at 0x236ab7c31c0>



In [6]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[6]: <matplotlib.image.AxesImage at 0x236ab8d0a60>



In [7]:
```
predictions=DeepFace.analyze(img)
```

```
Action: race: 100%|████████████████████████████████████████████████████
████████████████| 4/4 [00:08<00:00,  2.16s/it]
```

In [8]:
```
predictions
```

Out[8]:
```
[{'emotion': {'angry': 3.309897078640489e-08,
   'disgust': 2.7391443605494226e-11,
   'fear': 1.5320376078875597e-07,
   'happy': 99.9984622001648,
   'sad': 4.5461331921181625e-08,
   'surprise': 3.4626246758762136e-06,
   'neutral': 0.001533100294182077},
  'dominant_emotion': 'happy',
  'region': {'x': 58, 'y': 53, 'w': 107, 'h': 107},
  'face_confidence': 12.687342029239517,
  'age': 22,
  'gender': {'Woman': 13.832758367061615, 'Man': 86.16724610328674},
  'dominant_gender': 'Man',
  'race': {'asian': 0.004751507367473096,
   'indian': 0.07094774045981467,
   'black': 0.0002050057901215041,
   'white': 81.87616467475891,
   'middle eastern': 12.749972939491272,
   'latino hispanic': 5.297955870628357},
  'dominant_race': 'white'}]
```

In [9]:
```
data_list=predictions
```

In [10]:
```
data_dict = data_list[0]
```

In [11]:
```
type(data_dict)
```

Out[11]:
```
dict
```

In [12]:
```
data_dict['dominant_emotion']
```

Out[12]:
```
'happy'
```

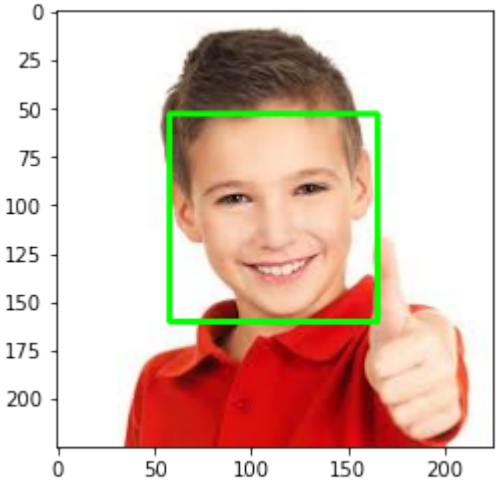
In [13]:
```
##we r tryin to draw rectanglr over face
```

In [14]:
```
faceCascade=cv2.CascadeClassifier(cv2.data.haarcascades+r"C:\Users\Pravalli
```

In [15]:
```
import cv2

# Specify the absolute path to the XML file
xml_path = r"C:\Users\Pravallika\Downloads\haarcascade_frontalface_default.

# Create CascadeClassifier object
faceCascade = cv2.CascadeClassifier(xml_path)

# Check if the cascade classifier is loaded successfully
if faceCascade.empty():
    print("Error: Cascade Classifier not loaded.")
else:
    # Read the image
    img = cv2.imread(r"C:\Users\Pravallika\OneDrive\Pictures\happyboy.jpg")

    # Check if the image is loaded successfully
    if img is None:
        print("Error: Image not loaded.")
    else:
        # Convert to grayscale
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # Detect faces
        faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeig

        # Draw rectangles around the faces
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

        # Display the result
        cv2.imshow("Detected Faces", img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

In [16]:
```
plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
```

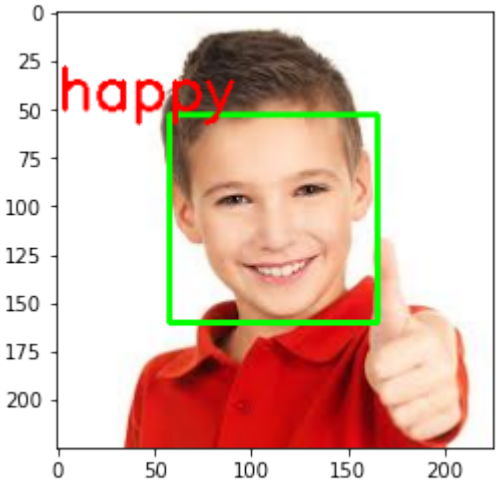
Out[16]:
```
<matplotlib.image.AxesImage at 0x236b7dab6d0>
```



In [17]:
```
font=cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,data_dict["dominant_emotion"],(0,50),font,1,(0,0,255),2,cv2
```

In [18]:
```
plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x236b7f45250>
```



In [ ]:

```
import cv2
from deepface import DeepFace

# Load the pre-trained face cascade
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_fr

# Start capturing video from the webcam
cap = cv2.VideoCapture(1)
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Cannot open webcam")

while True:
    # Read a frame from the video stream
    ret, frame = cap.read()

    if not ret:
        print("Failed to capture frame")
        break

    # Perform face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, 1.1, 4)

    # Check if faces are detected
    if len(faces) > 0:
        # Analyze emotions using DeepFace only if faces are detected
        results = DeepFace.analyze(frame, actions=['emotion'], enforce_dete

        # Extract dominant emotion from the first face (assuming there is o
        dominant_emotion = results[0]['dominant_emotion']

        # Draw rectangles around detected faces
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        # Display the dominant emotion on the frame
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame, dominant_emotion, (10, 50), font, 1, (0, 0, 255)

    # Display the frame
    cv2.imshow('Demo video', frame)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture object and close all windows
cap.release()
cv2.destroyAllWindows()
```

In [ ]: