Computer Vision :

- Computer Vision is a field of AI that enables machines to derive meaningful information from digital images, videos and other visual inputs.
- For installing cv2,write "pip install opencv-python" in Anaconda Prompt
- Images are stored in pixels
- Color intensity range is from 0-255 (pixel ranges from 0-255)
- RGB are the primary colors...all colors are made combining these three colors...colors are combined using 'dstack' or 'depth stack'

In [2]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#Reading an image
img = cv2.imread(r"C:\Users\CTTC\Downloads\Darshan-Raval.jpeg")
```

In [3]:
```
img   #image stored in matrix format
```

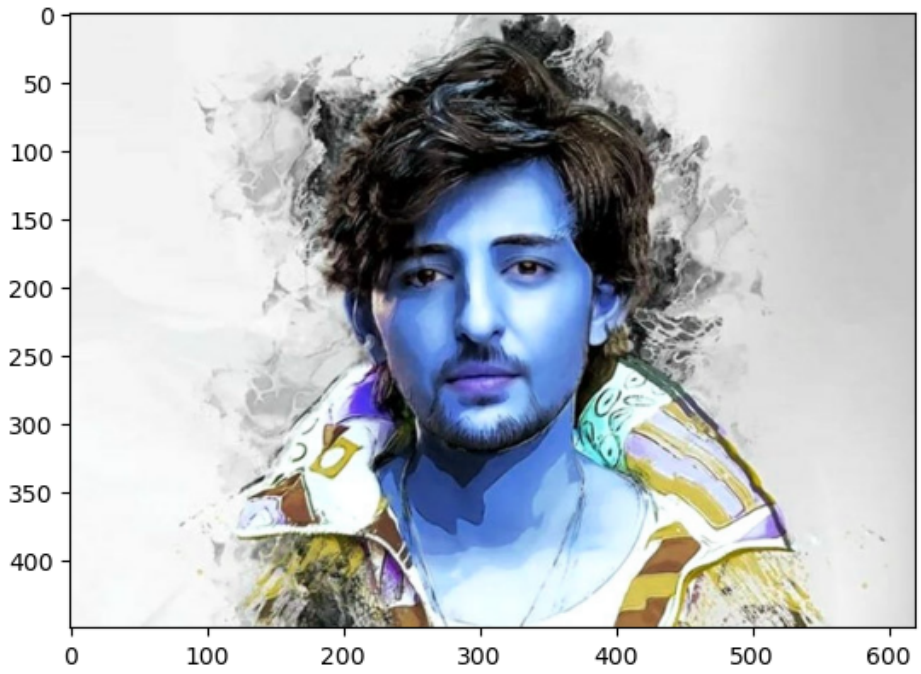
Out[3]:
```
array([[[239, 239, 239],
        [239, 239, 239],
        [239, 239, 239],
        ...,
        [188, 188, 188],
        [188, 188, 188],
        [188, 188, 188]],

       [[240, 240, 240],
        [240, 240, 240],
        [240, 240, 240],
        ...,
        [184, 184, 184],
        [184, 184, 184],
        [184, 184, 184]],

       [[240, 240, 240],
        [240, 240, 240],
        [240, 240, 240],
        ...,
        [184, 184, 184],
        [184, 184, 184],
        [184, 184, 184]],

       ...,

       [[248, 248, 248],
        [248, 248, 248],
        [248, 248, 248],
        ...,
        [217, 217, 217],
        [215, 215, 215],
        [215, 215, 215]],

       [[250, 250, 250],
        [249, 249, 249],
        [249, 249, 249],
        ...,
        [218, 218, 218],
        [216, 216, 216],
        [216, 216, 216]],

       [[250, 250, 250],
        [249, 249, 249],
        [249, 249, 249],
        ...,
        [219, 219, 219],
        [218, 218, 218],
        [217, 217, 217]]], dtype=uint8)
```

In [4]:
```
print(img.shape)
print(img.size)
print(img.ndim)
# (450,620,3) denotes
# 450,620 is the pixel size or shape of matrix
# 3 is for channel (3 for RGB and 1 for B/W images)
# size is product 450 x 620
#ndim is the dimension of array
```

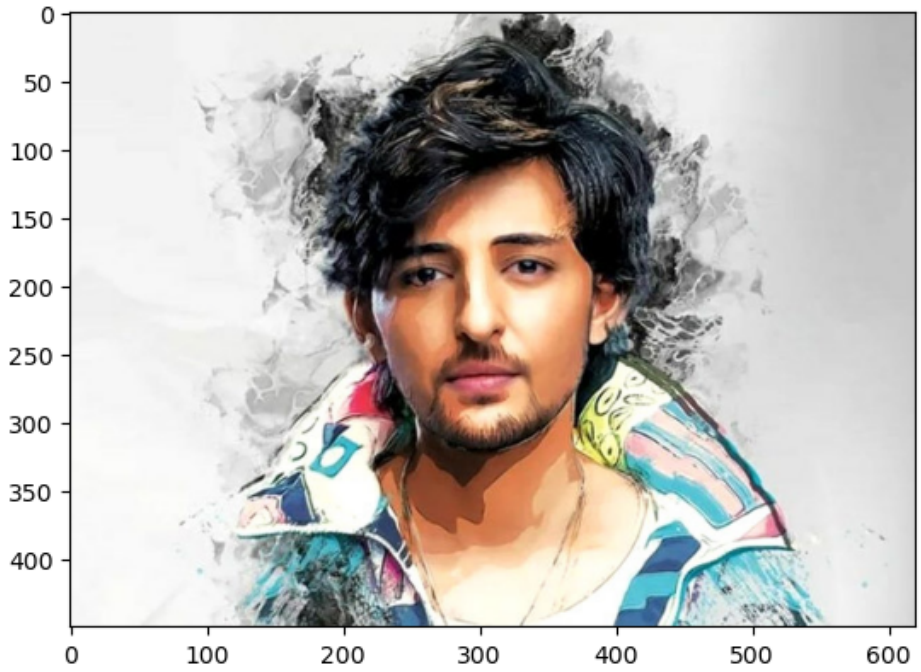
```
(450, 620, 3)
837000
3
```

In [5]:
```
#to show the image
cv2.imshow('Darshan',img)
#imshow() is used to show the image
cv2.waitKey(0)
#waitKey is used for, how much time the window will display or to hold the window for given time
#if waitKey is 0 press any key to close else it will be close in the given time(in miliseconds)
cv2.destroyAllWindows()
#used to destroy the opened window
```

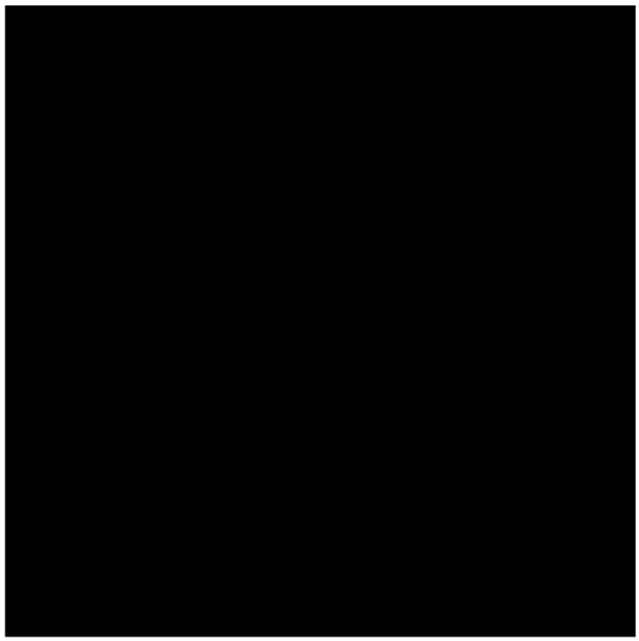
In [6]:
```
#to plot image
plt.imshow(img) #when we plot an image it stores it in BGR format bydefault
plt.show()
```



In [7]:
```
#convert color by using slicing method
plt.imshow(img[::,::,::-1])
plt.show()
```



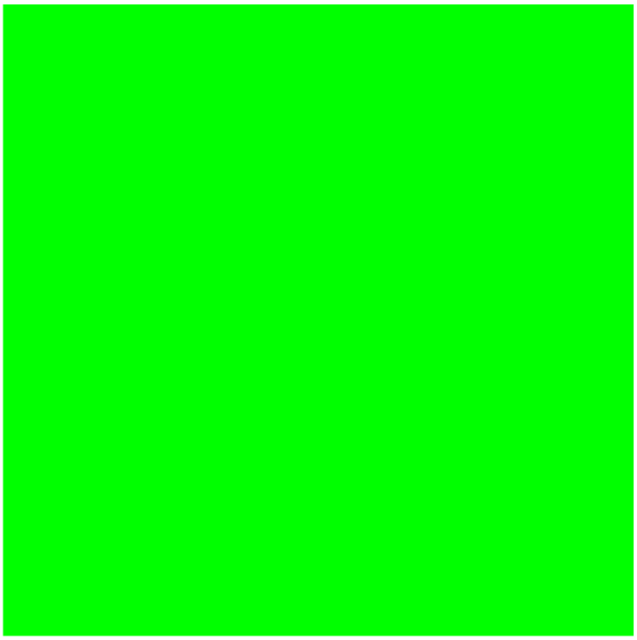
In [8]:
```
#color combinations
#black color
r = np.full((50,50),0)
g = np.full((50,50),0)
b = np.full((50,50),0)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```



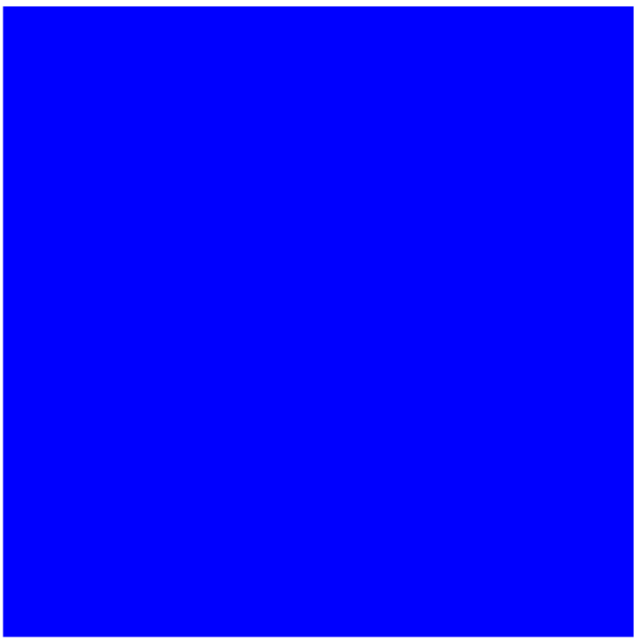
In [9]:
```
#white color
r = np.full((50,50),255)
g = np.full((50,50),255)
b = np.full((50,50),255)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```

In [10]:
```
#red
r = np.full((50,50),255)
g = np.full((50,50),0)
b = np.full((50,50),0)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```

In [11]:
```
#green
r = np.full((50,50),0)
g = np.full((50,50),255)
b = np.full((50,50),0)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```



In [12]:
```
#blue
r = np.full((50,50),0)
g = np.full((50,50),0)
b = np.full((50,50),255)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```



In [13]:
```
r = np.full((50,50),245)
g = np.full((50,50),10)
b = np.full((50,50),100)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```

In [14]:
```
#National Flag
#saffron
r = np.full((20,90),244)
g = np.full((20,90),120)
b = np.full((20,90),48)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```

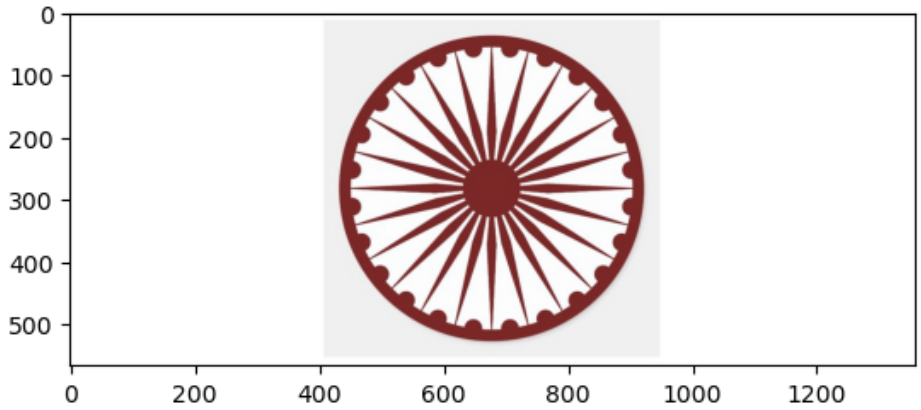
In [15]:
```
#white
r = np.full((20,90),255)
g = np.full((20,90),255)
b = np.full((20,90),255)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.axis('off')
plt.show()
```

In [16]:
```
#green
r = np.full((20,90),0)
g = np.full((20,90),102)
b = np.full((20,90),51)
rgb2 = np.dstack((r,g,b))
plt.imshow(rgb2)
plt.axis('off')
plt.show()
```

In [17]:
```
#National Flag
national_flag = np.vstack((rgb,rgb1,rgb2))
plt.imshow(national_flag)
plt.axis('off')
plt.show()
```

In [5]:
```
#National Flag with Ashok Chakra
#Ashok Chakra (by taking screen shot of img)
ac = cv2.imread(r"C:\Users\CTTC\Desktop\AI June\flag.jpg")  #path of the img
```

In [7]:
```
plt.imshow(ac)
plt.show()
```



In [8]:
```
ac1 = cv2.cvtColor(ac,cv2.COLOR_BGR2RGB)  #converted the color from BGR to RGB
plt.imshow(ac1)
plt.axis('off')
plt.show()
```

In [9]:
```
ac1.shape
```

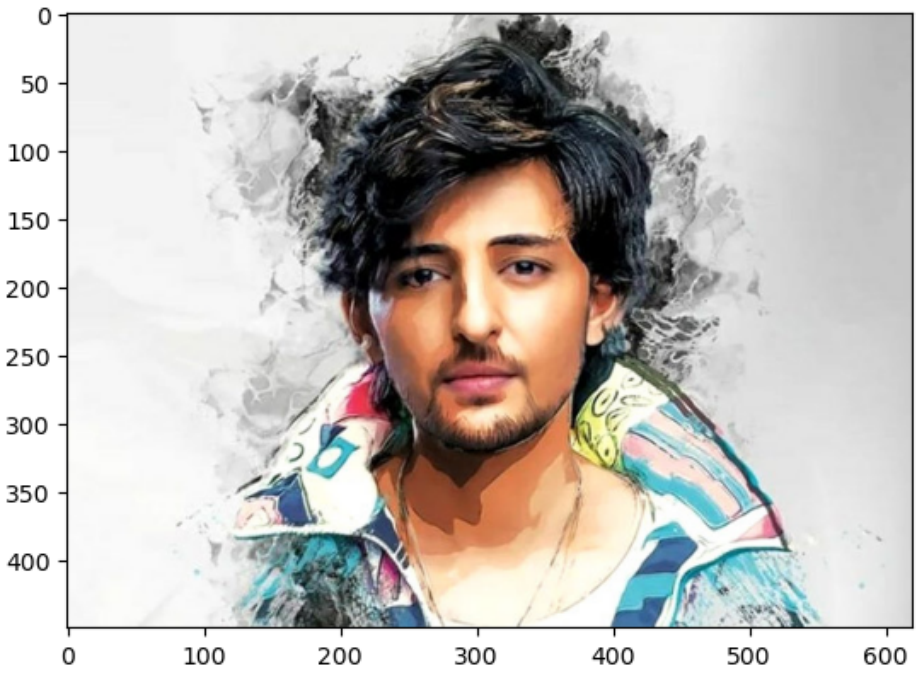
Out[9]: (566, 1359, 3)

In [10]:
```
#saffron (taken shape same as img)
r = np.full((566,1359),244)
g = np.full((566,1359),120)
b = np.full((566,1359),48)
rgb = np.dstack((r,g,b))
plt.imshow(rgb)
plt.axis('off')
plt.show()
```

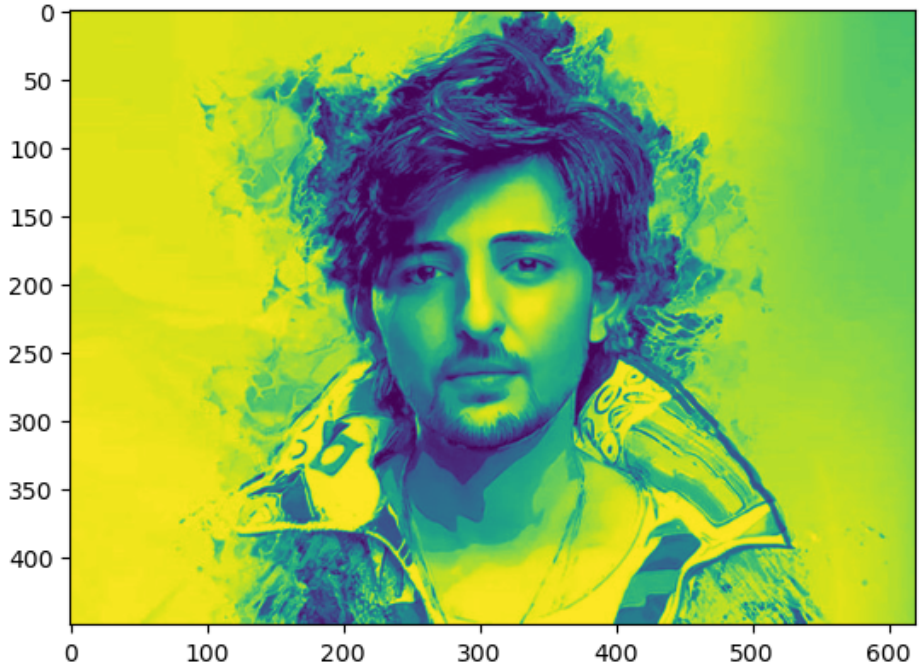
In [11]:
```
#green  (taken shape same as img)
r = np.full((566,1359),0)
g = np.full((566,1359),102)
b = np.full((566,1359),51)
rgb2 = np.dstack((r,g,b))
plt.imshow(rgb2)
plt.axis('off')
plt.show()
```

In [12]:
```
#National Flag
national_flag = np.vstack((rgb,ac1,rgb2)) #stacked the saffron,ashok chakra and green using vstacl
plt.imshow(national_flag)
plt.axis('off')
plt.show()
```

In [18]:
```
#BGR to RGB
rgb_img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(rgb_img)
plt.show()
```



In [19]:
```
#RGB to Grayscale
gray = cv2.cvtColor(rgb_img,cv2.COLOR_RGB2GRAY)
plt.imshow(gray)
#when we plot the img its appearance is like this as it takes the color in BGR bydeafault
plt.show()
```



In [20]:
```
cv2.imshow('Gray_Darshan',gray)  #cv2.imshow() will return us the gray img
cv2.waitKey(0)
cv2.destroyAllWindows()
```

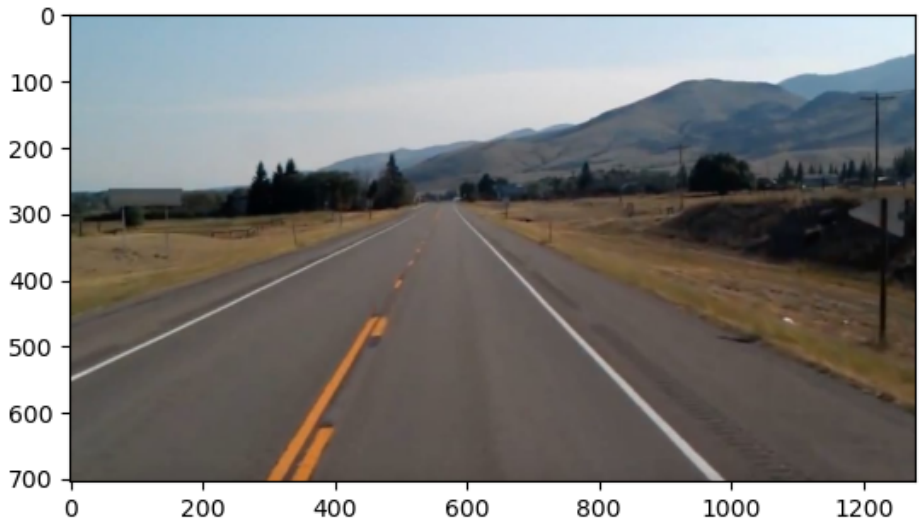
In [21]:
```
#Blur img
blur = cv2.GaussianBlur(gray,(7,7),5)
#(7,7) is the ksize or kernel size which convoles upon the given image which smoothens the noise
# this is why the img appears blur
#
cv2.imshow('Blur_Darshan',blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```
#capturing img from web cam
cap = cv2.VideoCapture(0)  #it opens camera and capture the images
if cap.isOpened():         # isOpened() checks if the camera is open or not
    ret,frame = cap.read()
    print(ret)             # ret : Return true if the camera is opened
    print(frame)           # frame : Return the matrix of image captured from webcam
else:
    ret = False

image = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.title("My Picture")
plt.axis('off')
plt.show()
cap.release()
```

In [22]:
```
#video feed
vdo = r"C:\Users\CTTC\Downloads\mixkit-waterfall-in-forest-2213-medium.mp4"

cv2.namedWindow(vdo)
cap = cv2.VideoCapture(vdo)
if cap.isOpened():
    ret,frame = cap.read()
    print(ret)                  # ret : Return true if the camera is opened
    print(frame)                # frame : Return the matrix of image captured from webcam
else:
    ret = False

while ret:
    ret,frame = cap.read()
    if not ret:
        break
    cv2.imshow(vdo,frame)
    if cv2.waitKey(1)==27: #27 is number of Esc or Escape key
        break
cv2.destroyAllWindows()
cap.release()
```

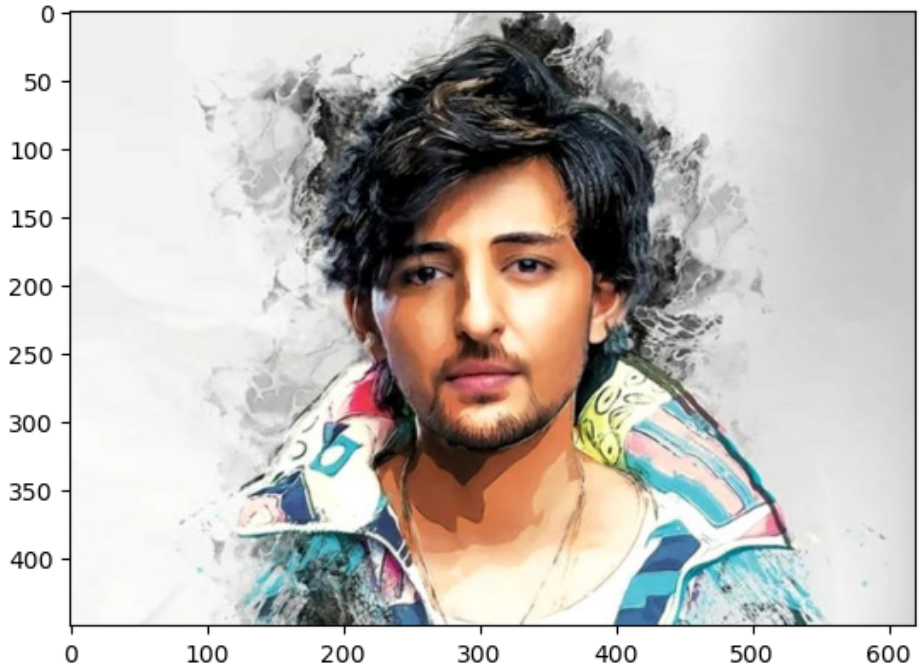
```
True
[[[104 128 125]
  [ 61  85  82]
  [ 44  68  65]
  ...
  [ 46  55  41]
  [ 42  51  37]
  [ 48  57  43]]

 [[ 31  55  52]
  [ 15  39  36]
  [ 36  60  57]
  ...
  [ 46  55  41]
  [ 49  58  44]
  [ 54  63  49]]

 [[ 25  49  46]
  [  1  25  22]
  [ 50  74  71]
  ...
  [ 46  55  41]
  [ 49  58  44]
  [ 48  57  43]]

 ...

 [[ 41  49  48]
  [ 41  49  48]
  [ 43  51  50]
  ...
  [ 31  37  32]
  [ 29  35  30]
  [ 29  35  30]]

 [[ 41  49  48]
  [ 41  49  48]
  [ 45  53  52]
  ...
  [ 29  35  30]
  [ 33  39  34]
  [ 33  39  34]]

 [[ 41  49  48]
  [ 45  53  52]
  [ 51  59  58]
  ...
  [ 32  38  33]
  [ 40  46  41]
  [ 43  49  44]]]
```

In [23]:
```
#Edge Detection
road = cv2.imread(r"C:\Users\CTTC\Downloads\WhatsApp Image 2023-06-04 at 8.10.07 PM.jpeg")
plt.imshow(road[::,::,::-1])
plt.show()
```

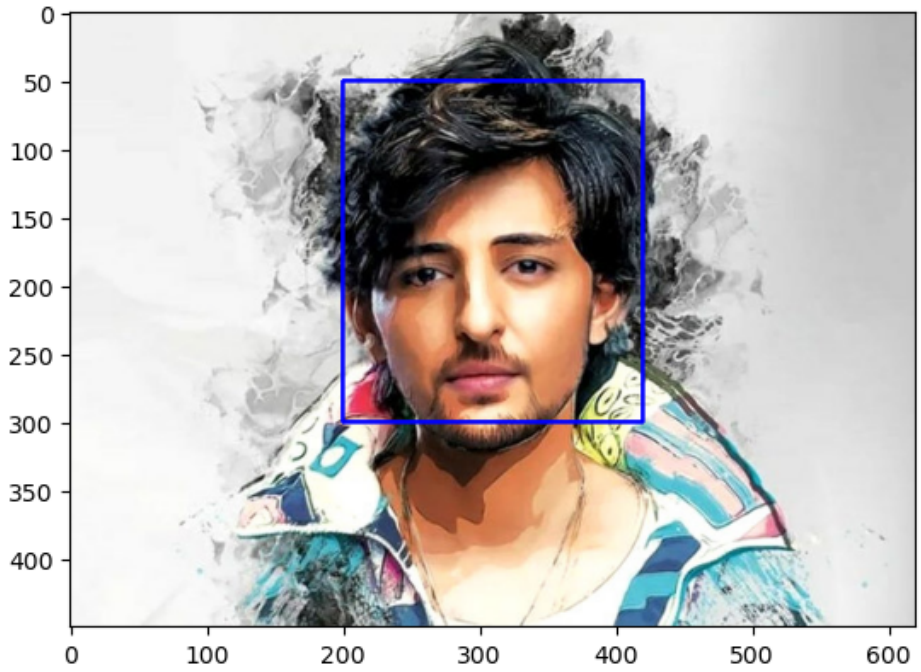


In [24]:
```
#Canny Edge Detection
road = cv2.imread(r"C:\Users\CTTC\Downloads\WhatsApp Image 2023-06-04 at 8.10.07 PM.jpeg")
edge_detc = cv2.Canny(road,30,120)
cv2.imshow('Canny',edge_detc)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [25]:
```
#face detection
plt.imshow(rgb_img)
plt.show()
```



In [26]:
```
fimg = cv2.rectangle(rgb_img,(200,300),(420,50),(0,0,255),thickness=2)
plt.imshow(fimg)
plt.show()
```



In [27]:
```
#face detection using haarcascade
face_data = r"C:\Users\CTTC\Downloads\haarcascades\haarcascade_frontalface_default.xml"
model = cv2.CascadeClassifier(face_data)

img = cv2.imread(r"C:\Users\CTTC\Downloads\Darshan-Raval.jpeg")

facepoints = model.detectMultiScale(img)
#detectMultiScale is used to identify more than one faces in an image

for x,y,w,h in facepoints:
    cv2.rectangle(img,(x,y),(x+w,y+h),(123,45,20),thickness=3)
cv2.imshow('Image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```
#face detection using webcam
#it will capture video and will detect multiple faces
#colored images will be collected
face_detect = r"C:\Users\CTTC\Downloads\haarcascades\haarcascade_frontalface_default.xml"
cap = cv2.VideoCapture(0)

ret,frame = cap.read()

while ret:
    ret,image = cap.read()
    print(ret)
    print(image)

    if not ret:
        break
    model = cv2.CascadeClassifier
    (face_detect)
    face = model.detectMultiScale(image)

    for x,y,w,h in face:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)

    cv2.imshow("Image",image)
    if cv2.waitKey(10)==27:
        break
cv2.destroyAllWindows()
cap.release()
```

In [ ]:
```
#face detection using webcam
#it will capture video and will detect multiple faces
#video will capture in B/W
face_detect = r"C:\Users\CTTC\Downloads\haarcascades\haarcascade_frontalface_default.xml"
cap = cv2.VideoCapture(0)

ret,frame = cap.read()

while ret:
    ret,image = cap.read()
    print(ret)
    print(image)

    if not ret:
        break
    model = cv2.CascadeClassifier
    (face_detect)
    face = model.detectMultiScale(image)

    for x,y,w,h in face:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
    img1 = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)
    cv2.imshow("Image",img1)
    if cv2.waitKey(10)==27:
        break
cv2.destroyAllWindows()
cap.release()
```