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

Load image

In [148…
```
def read_file(filename):
    img=cv.imread(filename)
    img=cv.cvtColor(img , cv.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.axis("off")
    plt.title("IMAGE")
    plt.show()
    return img

filename=(r"C:\Users\asif\Downloads\dhoni.jpg")
img = read_file(filename)
org_img=np.copy(img)

#CREATE EDGE MASK

def edge_mask(img,line_size,blur_value):
    gray=cv.cvtColor(img,cv.COLOR_RGB2GRAY)
    gray_blur=cv.medianBlur(gray,blur_value)
    edges=cv.adaptiveThreshold (gray_blur,255,cv.ADAPTIVE_THRESH_MEAN_C,cv.THRESH_B
    return edges
line_size,blur_value=5,5
edges=edge_mask(img,line_size,blur_value)
plt.imshow(edges,cmap="binary")
plt.axis("off")
plt.title("IMAGE WITH EDGE MASK")
plt.show()

# REDUCE THE COLOR PALATTE

def color_quantization(img,k):
    # Transform the image
    data = np.float32(img).reshape((-1,3))
    # Determine Criteria
    criteria = (cv.TERM_CRITERIA_EPS+ cv.TERM_CRITERIA_MAX_ITER,20,0.001)
    # Implementing K-Means
    ret,label,centre=cv.kmeans(data,k,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)
    centre = np.uint8(centre)
    result = centre[label.flatten()]
    result = result.reshape(img.shape)
    return result
img_quantiz = color_quantization(img,k=2)
plt.imshow(img_quantiz)
plt.axis("off")
plt.title("COLOR PALATTE IMAGE")
plt.show()

# REDUCE THE NOISE

blurred = cv.bilateralFilter(img_quantiz,d=5,sigmaColor=300,sigmaSpace=300)
plt.imshow(blurred)
plt.axis("off")
plt.title("NOISE REDUCE IMAGE")
plt.show()

# COMBINING EDGE MASK WITH THE QUANTIZ IMAGE

def cartoon():
    c=cv.bitwise_and(blurred,blurred,mask=edges)
    plt.imshow(c)
    plt.axis("off")
    plt.title("CARTOONIFIED IMAGE")
    plt.show()
    plt.imshow(org_img)
    plt.axis("off")
    plt.title("ORIGINAL IMAGE")
    plt.show()
cartoon()
```