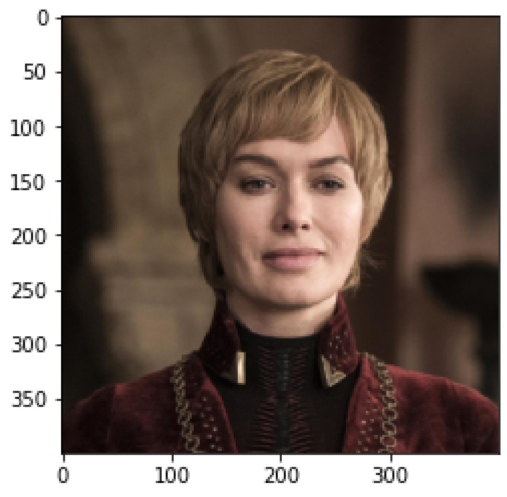
In [28]:

```python
# Loading Libraries
import cv2
import matplotlib.pyplot as plt
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_alt.xml")
#eyeCascade = cv2.CascadeClassifier("haarcascade_eye.xml")
```

In [29]:

```python
# Loading Images
face = cv2.cvtColor(cv2.imread('cersie.png'),cv2.COLOR_BGR2RGB)
face = cv2.resize(face,(400,400))
glasses = cv2.cvtColor(cv2.imread('glasses.png'),cv2.COLOR_BGR2RGB)
plt.imshow(face)
```

```
<matplotlib.image.AxesImage at 0x17262d72fd0>
```



```
In [30]:
```

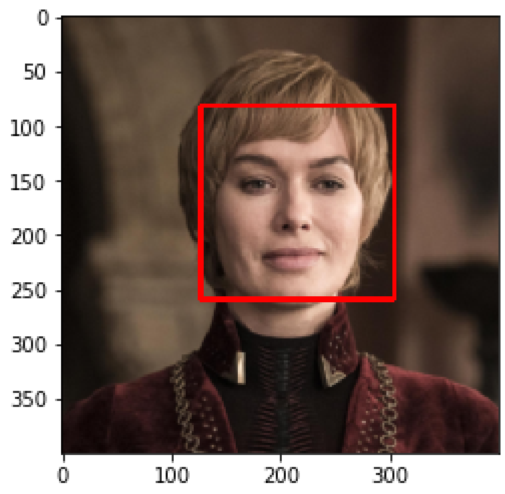
```
# Visualizing - Tyrion
face_copy = face.copy()
faces = faceCascade.detectMultiScale(face)
if len(faces) > 0:
    print("Face DETECTED: ",faces)

    faces = sorted(faces, key=lambda f:f[2]*f[3])

    for x,y,w,h in faces[-1:]:
        cv2.rectangle(face_copy,(x,y),(x+w,y+h),(255,0,0),3)
    plt.imshow(face_copy)

else:
    print("Face NOT DETECTED!! ")
```

```
Face DETECTED:  [[127  82 177 177]]
```



In [31]:

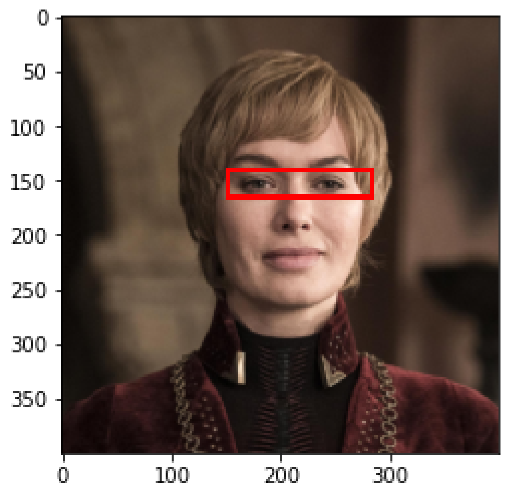
```
# Eyes Approx Detection
face_copy = face.copy()
x,y,w,h = faces[-1]
print(x,y,w,h)

ly = int(y+h/3)
lx = int(x+w/7)
ry = int(ly+h/7)
rx = int(lx+(3*w)/4)

cv2.rectangle(face_copy,(lx,ly),(rx,ry),(255,0,0),3)
plt.imshow(face_copy)
```

```
127 82 177 177
```

```
<matplotlib.image.AxesImage at 0x1725c5d67f0>
```



In [32]:

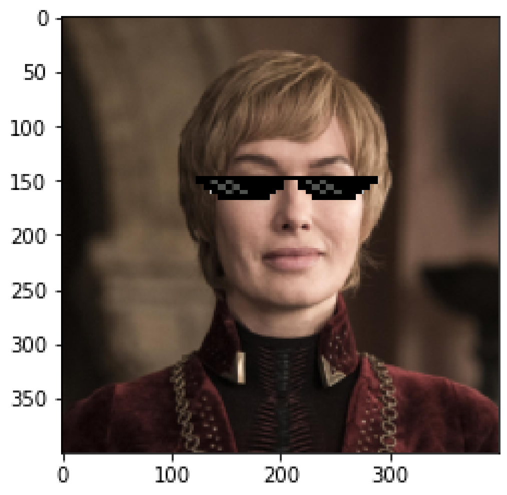
```
face_copy = face.copy()
glass_width = int((rx-lx)*1.3)
glass_height = int((ry-ly)*2.5)
print(glass_width,glass_height)

try:
    glasses = cv2.resize(glasses,(glass_width,glass_height))

    for i in range(glass_height):
        for j in range(glass_width):
            for k in range(3):
                if glasses[i][j][k]<235: #avoiding white
                    face_copy[int(ly*.9)+i][int(lx*.8)+j][k] = glasses[i][j][k]

    plt.imshow(face_copy)
except:
    print("Error Occured")
    print("Glass Width",glass_width)
    print("Glass Height",glass_height)
```

```
171 62
```



In [36]:

```
def put_glasses(glasses,frame,x,y,w,h):
    ly = int(y+h/3)
    lx = int(x+w/7)
    ry = int(ly+h/7)
    rx = int(lx+(3*w)/4)
    glass_width = int((rx-lx)*1.3)
    glass_height = int((ry-ly)*2.5)

    glasses = cv2.resize(glasses,(glass_width,glass_height))
    #cv2.rectangle(frame,(lx,ly),(rx,ry),(255,0,0),3)

    for i in range(glass_height):
        for j in range(glass_width):
            for k in range(3):
                if glasses[i][j][k]<235: #avoiding white
                    frame[int(ly*.9)+i][int(lx*.9)+j][k] = glasses[i][j][k]

    return frame
```

In [37]:

```
video_capture = cv2.VideoCapture(0)
```

In [38]:
```
while True:

    # Capture frame-by-frame
    ret, frame = video_capture.read()

    if ret == False:
        continue

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(

    # Draw a rectangle around the faces
    for x, y, w, h in faces:
        #cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        #cv2.putText(frame,"Person Detected",(x,y),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255
        frame = put_glasses(glasses,frame,x,y,w,h)


    cv2.imshow('Video', frame)


    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

In [ ]: