In [2]:
```
# import necessary libraries:
import os
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

# preparing image data generator with minor data augmentations to keep the training time
train_datagen = ImageDataGenerator(rescale = 1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')


test_datagen = ImageDataGenerator( rescale = 1.0/255)

train_generator = train_datagen.flow_from_directory('/Users/ASUSvB/Desktop/archive/Datas
                                                    batch_size =128 ,
                                                    class_mode = 'binary',
                                                    target_size = (64, 64))

validation_generator =  test_datagen.flow_from_directory('/Users/ASUSvB/Desktop/archive/
                                                         batch_size  = 128,
                                                         class_mode  = 'binary',
                                                         target_size = (64, 64))
```

```
Found 160000 images belonging to 2 classes.
Found 22598 images belonging to 2 classes.
```

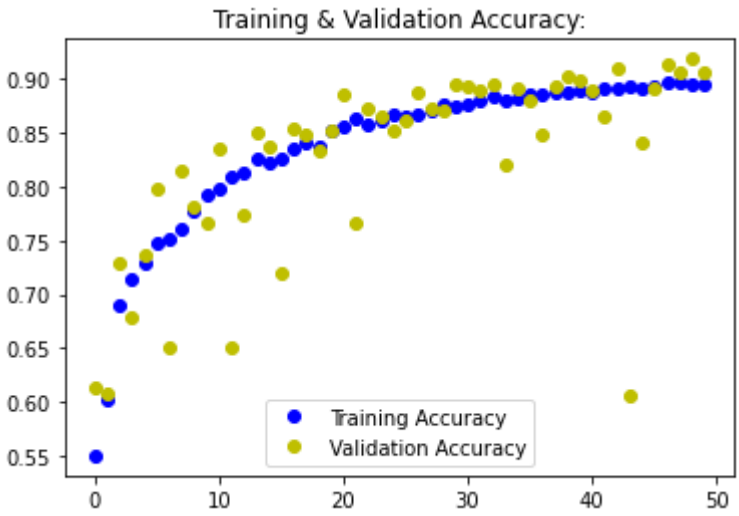
In [ ]:
```
# 160000 images for the model to pick up the local patterns
```

In [3]:

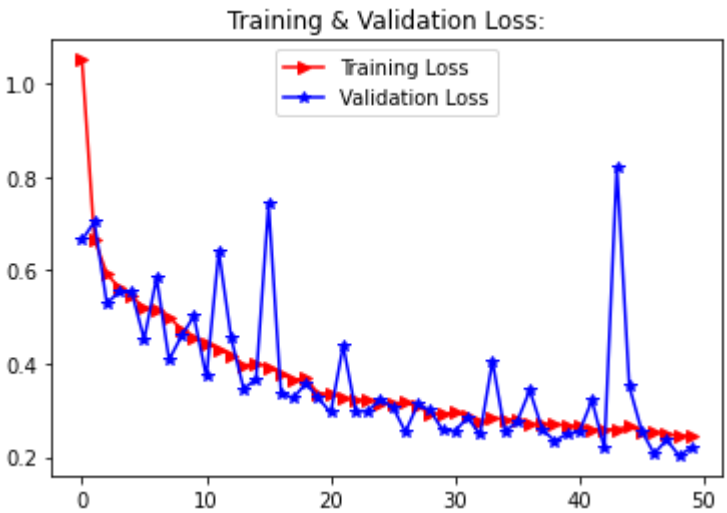
```
# building the model:

from keras.optimizers import Adam
model = tf.keras.models.Sequential([
    # 1st conv
  tf.keras.layers.Conv2D(96, (11,11),strides=(4,4), activation='relu', input_shape=(64,
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2,2)),
    # 2nd conv
  tf.keras.layers.Conv2D(256, (11,11),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
     # 3rd conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 4th conv
  tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
    # 5th Conv
  tf.keras.layers.Conv2D(256, (3, 3), strides=(1, 1), activation='relu',padding="same"),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.MaxPooling2D(2, strides=(2, 2)),
  # To Flatten layer
  tf.keras.layers.Flatten(),
  # To FC layer 1
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  #To FC layer 2
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(1, activation='sigmoid')
  ])
model.compile(
    optimizer=Adam(lr=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
   )

#Model Training using model.fit:
hist = model.fit(train_generator,steps_per_epoch=128,epochs=50,validation_data=validatio

# 160000 images for the model to pick up the local patterns
```

```
0.8940 - val_loss: 0.8220 - val_accuracy: 0.6059
Epoch 45/50
128/128 [==============================] - 364s 3s/step - loss: 0.2658 - accuracy:
0.8906 - val_loss: 0.3510 - val_accuracy: 0.8406
Epoch 46/50
128/128 [==============================] - 361s 3s/step - loss: 0.2524 - accuracy:
0.8930 - val_loss: 0.2560 - val_accuracy: 0.8916
Epoch 47/50
128/128 [==============================] - 370s 3s/step - loss: 0.2529 - accuracy:
0.8960 - val_loss: 0.2066 - val_accuracy: 0.9136
Epoch 48/50
128/128 [==============================] - 374s 3s/step - loss: 0.2498 - accuracy:
0.8967 - val_loss: 0.2372 - val_accuracy: 0.9055
Epoch 49/50
128/128 [==============================] - 383s 3s/step - loss: 0.2444 - accuracy:

0.8959 - val_loss: 0.2026 - val_accuracy: 0.9188
Epoch 50/50
128/128 [==============================] - 702s 5s/step - loss: 0.2456 - accuracy:
0.8942 - val loss: 0.2201 - val accuracy: 0.9055
```

In [123]:
```
# plot the training vs validation loss and accuracy graphs:

import matplotlib.pyplot as plt
acc = hist.history['accuracy']
val_acc = hist.history['val_accuracy']
loss = hist.history['loss']
val_loss = hist.history['val_loss']
epochs = range(len(acc))

# plot Training & Validation Accuracy:
plt.plot(epochs, acc, 'bo', label='Training Accuracy')
plt.plot(epochs, val_acc, 'yo', label='Validation Accuracy')
plt.title('Training & Validation Accuracy:')
plt.legend(loc='lower center')
plt.figure()
plt.show()

# plot Training & Validation Loss:
plt.plot(epochs, loss, 'r', marker='>', label='Training Loss')
plt.plot(epochs, val_loss, 'b', marker='*', label='Validation Loss')
#plt.plot(epochs, loss, 'ro', label='Training Loss')
#plt.plot(epochs, val_loss, 'bo', label='Validation Loss')
plt.title('Training & Validation Loss:')
plt.legend(loc='upper center')
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

In [111]:

```
from tkinter import *
from PIL import ImageTk, Image
from tkinter import filedialog
import os

import numpy as np
import matplotlib.pyplot as plt

from keras.preprocessing import image

root = Tk()
root.geometry("800x500+500+300")
root.resizable(width=True, height=True)

def openfn():
    filename = filedialog.askopenfilename(title='Browse a photo')
    return filename
def open_imgg():
    g = openfn()
    imgg = Image.open(g)
    img = Image.ANTIALIAS
    imgg = ImageTk.PhotoImage(imgg)
    panel = Label(root, image = imgg)
    panel.image = imgg
    panel.pack()

      #Label(win, text="The File is located at : " + str(path), font=('Aerial 11')).pack

# Add a Label widget
#label = Label(win, text="Click the Button to browse the Files", font=('Georgia 13'))
#label.pack(pady=10)
  # file = filedialog.askopenfile(mode='r', filetypes=[('Python Files', '*.py')])


btn = Button(root, text='Browse a photo', command=open_imgg).pack()

root.mainloop()

#img = image.load_img(open_imgg, target_size=(64, 64))

# predicting images
#path = "/Users/ASUSvB/Desktop/archive/Dataset/Test/Female/160320.jpg"
#img = image.load_img(path, target_size=(64, 64))
#img = image.load_img(open_img, target_size=(64, 64))
```

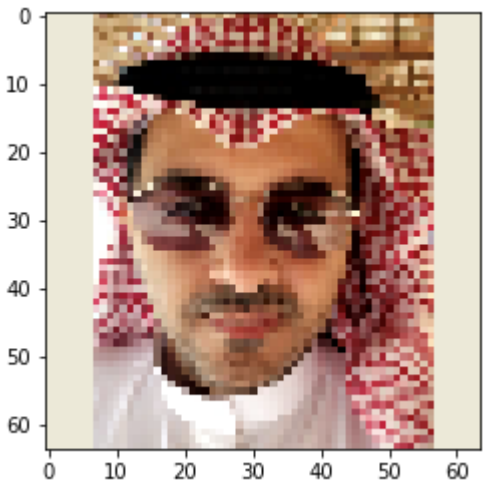
In [110]:

```
# Import the required Libraries
from tkinter import *
from tkinter import ttk, filedialog
from tkinter.filedialog import askopenfile
import os

# Create an instance of tkinter frame
win = Tk()

# Set the geometry of tkinter frame
win.geometry("800x500")

def open_file():
   file = filedialog.askopenfile(title='open')

   if file:
      path = os.path.abspath(file.name)
      Label(win, text="The File is located at : " + str(path), font=('Aerial 11')).pack(

# Add a Label widget
label = Label(win, text="Click the Button load a photo", font=('Georgia 13'))
label.pack(pady=10)

# Create a Button
ttk.Button(win, text="Browse a photo", command=open_file).pack(pady=20)

win.mainloop()

ppp = path
  # file = filedialog.askopenfile(mode='r', filetypes=[('Python Files', '*.py')])
```

In [122]:
```
import numpy as np
from keras.preprocessing import image

# Testing, the follwing image is mine and it is copletly not form the dataset:
path = "/Users/ASUSvB/Desktop/archive/Dataset/Test/Male/333.png"

# Testing, the follwing image is Not from the dataset completly new:
#path = "/Users/ASUSvB/Desktop/archive/Dataset/Test/Female/555.jpg"

img = image.load_img(path, target_size=(64, 64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)

# making decision
images = np.vstack([x])
classes = model.predict(images, batch_size=1)
print(classes[0])
if classes[0]> 0.5:
    print("-It is very likely he is Male.    .البرنامج يقول أنه رجل-")   #showing the decision
elif classes[0] <= 0.5:
    print("-It is very likely she is Female.    .البرنامج يقول أنها أنثى-")  #showing the decisi
else:
    print( "Error!")

# showing the tested image
plt.imshow(img)
```

```
[1.]
-It is very likely he is Male.    .البرنامج يقول أنه رجل-
```

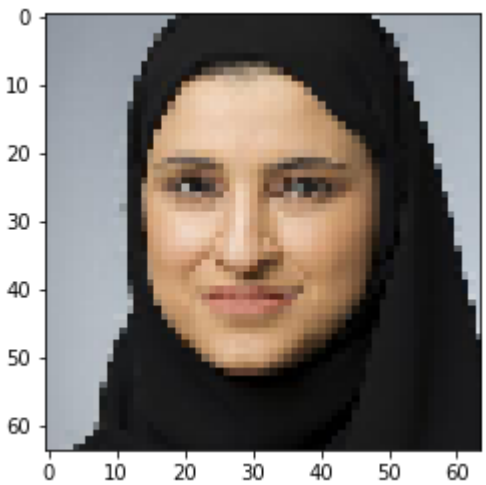
Out[122]:
```
<matplotlib.image.AxesImage at 0x185d6d8c490>
```



In [121]:
```
import numpy as np
from keras.preprocessing import image

# Testing, the follwing image is mine and it is copletly not form the dataset:
#path = "/Users/ASUSvB/Desktop/archive/Dataset/Test/Male/999.png"

# Testing, the follwing image is Not from the dataset completly new:
path = "/Users/ASUSvB/Desktop/archive/Dataset/Test/Female/555.jpg"

img = image.load_img(path, target_size=(64, 64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)

# making decision
images = np.vstack([x])
classes = model.predict(images, batch_size=1)
print(classes[0])
if classes[0]> 0.5:
    print("-It is very likely he is Male.    .البرنامج يقول أنه رجل-")   #showing the decision
elif classes[0] <= 0.5:
    print("-It is very likely she is Female.    .البرنامج يقول أنها أنثى-")  #showing the decisi
else:
    print( "Error!")

# showing the tested image
plt.imshow(img)
```

```
[2.0179896e-13]
-It is very likely she is Female.    .البرنامج يقول أنها أنثى-
```

Out[121]:
```
<matplotlib.image.AxesImage at 0x185d69492b0>
```



In [ ]: