In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
 shear_range = 0.2,
 zoom_range = 0.2,
 horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [2]:
```
test_set = test_datagen.flow_from_directory('Dataset/Test',
 target_size = (224, 224),
 batch_size = 32,
 class_mode = 'categorical')
```

```
Found 134 images belonging to 2 classes.
```

In [3]:
```
from tensorflow.keras.models import load_model

# Model saved with Keras model.save()
MODEL_PATH ='model_vgg19.h5'

# Load your trained model
model = load_model(MODEL_PATH)
```

In [4]:
```
 y_pred = model.predict(test_set)
```

In [5]:
```
y_pred
```

Out[5]:
```
array([[3.51420522e-01, 6.48579478e-01],
       [3.15008223e-01, 6.84991777e-01],
       [2.49052513e-02, 9.75094676e-01],
       [8.44473004e-01, 1.55527011e-01],
       [9.99973655e-01, 2.63286001e-05],
       [9.99937415e-01, 6.25946559e-05],
       [1.16299026e-01, 8.83701026e-01],
       [2.67133638e-02, 9.73286688e-01],
       [9.64657664e-01, 3.53423730e-02],
       [9.99940872e-01, 5.91624339e-05],
       [2.26018913e-02, 9.77398098e-01],
       [7.79348314e-02, 9.22065198e-01],
       [4.65420596e-02, 9.53457952e-01],
       [8.48463356e-01, 1.51536673e-01],
       [9.99068439e-01, 9.31616058e-04],
       [7.60960579e-02, 9.23903942e-01],
       [9.80807245e-01, 1.91927906e-02],
       [9.93485034e-01, 6.51501724e-03],
       [5.64740658e-01, 4.35259372e-01],
       [1.13631934e-01, 8.86368096e-01],
       [9.99504447e-01, 4.95534972e-04],
       [9.99584496e-01, 4.15479502e-04],
       [1.78368106e-01, 8.21631908e-01],
       [7.74592161e-01, 2.25407809e-01],
       [9.96749878e-01, 3.25014745e-03],
       [9.33516979e-01, 6.64829910e-02],
       [8.48820686e-01, 1.51179254e-01],
       [9.98122752e-01, 1.87728333e-03],
       [1.00000000e+00, 5.56785418e-09],
       [9.99680638e-01, 3.19386541e-04],
       [9.92535949e-01, 7.46406196e-03],
       [7.08026171e-01, 2.91973799e-01],
       [1.79526001e-01, 8.20473969e-01],
       [9.99854922e-01, 1.45101134e-04],
       [9.97438550e-01, 2.56145676e-03],
       [9.98226702e-01, 1.77325646e-03],
       [1.33023798e-01, 8.66976261e-01],
       [9.89062250e-01, 1.09377559e-02],
       [1.24900743e-01, 8.75099301e-01],
       [3.01815663e-02, 9.69818354e-01],
       [9.51245964e-01, 4.87540960e-02],
       [2.19181478e-02, 9.78081822e-01],
       [9.99999285e-01, 6.78014885e-07],
       [9.52461421e-01, 4.75385636e-02],
       [3.86598147e-02, 9.61340129e-01],
       [9.99790609e-01, 2.09422928e-04],
       [9.98033464e-01, 1.96657726e-03],
       [8.12208176e-01, 1.87791809e-01],
       [6.58888638e-01, 3.41111332e-01],
       [4.80751060e-02, 9.51924920e-01],
       [3.90607625e-01, 6.09392405e-01],
       [9.99646425e-01, 3.53570882e-04],
       [2.56446630e-01, 7.43553400e-01],
       [9.99147534e-01, 8.52417143e-04],
       [2.90788934e-02, 9.70921099e-01],
       [7.33568192e-01, 2.66431868e-01],
       [7.25330263e-02, 9.27466989e-01],
```

```
[7.85597324e-01, 2.14402705e-01],
[8.54977012e-01, 1.45022959e-01],
[1.81540042e-01, 8.18459928e-01],
[2.82993298e-02, 9.71700609e-01],
[4.82946366e-01, 5.17053604e-01],
[6.69995770e-02, 9.33000445e-01],
[9.99913931e-01, 8.60950167e-05],
[9.26610827e-01, 7.33892098e-02],
[9.99835610e-01, 1.64366837e-04],
[8.14394414e-01, 1.85605556e-01],
[3.09107453e-02, 9.69089270e-01],
[4.96625096e-01, 5.03374934e-01],
[9.99930739e-01, 6.92849644e-05],
[6.21868186e-02, 9.37813222e-01],
[2.77041905e-02, 9.72295761e-01],
[9.99862194e-01, 1.37764408e-04],
[9.99998689e-01, 1.34526908e-06],
[1.42495185e-01, 8.57504845e-01],
[9.99980807e-01, 1.92336047e-05],
[9.69726026e-01, 3.02740131e-02],
[4.46416214e-02, 9.55358326e-01],
[8.46040919e-02, 9.15395975e-01],
[1.37733385e-01, 8.62266600e-01],
[9.99998569e-01, 1.47930052e-06],
[9.90637481e-01, 9.36260261e-03],
[1.44461364e-01, 8.55538666e-01],
[9.99994755e-01, 5.23468771e-06],
[8.15682530e-01, 1.84317455e-01],
[3.77726299e-03, 9.96222734e-01],
[9.96966064e-01, 3.03391251e-03],
[9.99985337e-01, 1.46407219e-05],
[7.20607400e-01, 2.79392660e-01],
[9.78302777e-01, 2.16971543e-02],
[5.22698089e-02, 9.47730184e-01],
[9.55748677e-01, 4.42513563e-02],
[9.97366607e-01, 2.63345707e-03],
[4.26607013e-01, 5.73393047e-01],
[9.37498827e-03, 9.90625024e-01],
[9.99096274e-01, 9.03777895e-04],
[1.34621561e-02, 9.86537814e-01],
[9.99836445e-01, 1.63586170e-04],
[9.04192328e-01, 9.58076790e-02],
[5.62469661e-01, 4.37530398e-01],
[9.11167204e-01, 8.88328403e-02],
[1.99335113e-01, 8.00664902e-01],
[9.99961019e-01, 3.89583693e-05],
[7.92534769e-01, 2.07465261e-01],
[9.99957442e-01, 4.25866310e-05],
[9.99995351e-01, 4.68705957e-06],
[7.36469626e-02, 9.26352978e-01],
[2.93028634e-02, 9.70697105e-01],
[6.52000666e-01, 3.47999364e-01],
[9.99721229e-01, 2.78760330e-04],
[9.99995708e-01, 4.34720141e-06],
[8.01913381e-01, 1.98086604e-01],
[5.37910238e-02, 9.46208954e-01],
[9.99727428e-01, 2.72591918e-04],
```

```
       [1.02938088e-02, 9.89706159e-01],
       [9.89815593e-01, 1.01844380e-02],
       [2.15925537e-02, 9.78407443e-01],
       [9.75712121e-01, 2.42878925e-02],
       [9.94786859e-01, 5.21314982e-03],
       [1.04434900e-02, 9.89556491e-01],
       [2.66673174e-02, 9.73332644e-01],
       [9.99927998e-01, 7.19879099e-05],
       [9.72831368e-01, 2.71686390e-02],
       [7.95964450e-02, 9.20403600e-01],
       [1.00000000e+00, 1.31814044e-08],
       [6.61823452e-01, 3.38176608e-01],
       [8.65602374e-01, 1.34397611e-01],
       [9.99700785e-01, 2.99298030e-04],
       [9.81848955e-01, 1.81510355e-02],
       [9.53926682e-01, 4.60733287e-02],
       [9.99316454e-01, 6.83533610e-04],
       [6.72240779e-02, 9.32775974e-01],
       [9.41605926e-01, 5.83940633e-02],
       [9.13027167e-01, 8.69727731e-02]], dtype=float32)
```

In [6]:
```
import numpy as np
y_pred=np.argmax(y_pred, axis=1)
```

In [57]:
```
y_pred[48:]
```

Out[57]:
```
array([0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
      dtype=int64)
```

In [8]:
```
test_set.class_indices
```

Out[8]:
```
{'Parasite': 0, 'Uninfected': 1}
```

In [56]:
```
y_true=test_set.classes
y_true[48:]
```

Out[56]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [58]:
```
from sklearn.metrics import confusion_matrix
cf_matrix =confusion_matrix(y_pred[48:],y_true[48:])
```

In [59]:
```
group_names = ["True Infected","False Uninfected","False Infected","True Uninf
ected"]
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
```

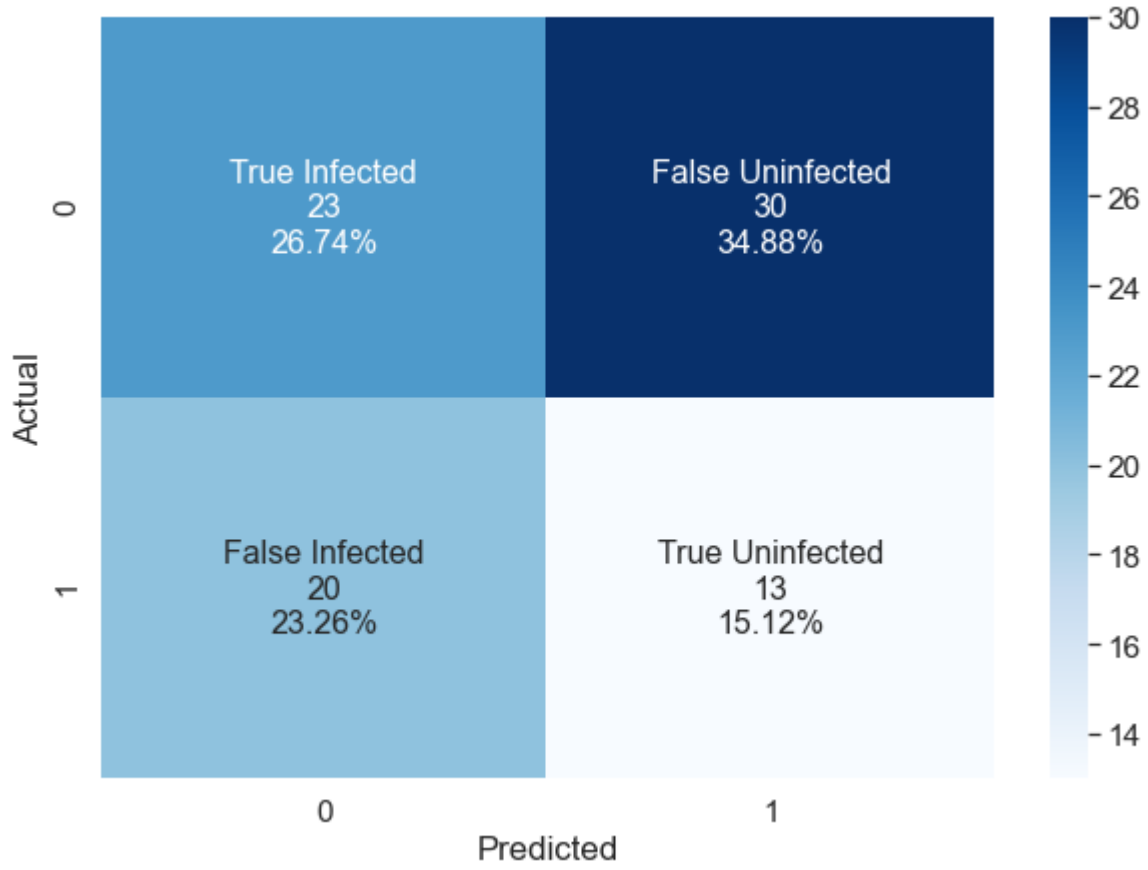
In [60]:
```
labels = np.asarray(labels).reshape(2,2)
print(labels)
```

```
[['True Infected\n23\n26.74%' 'False Uninfected\n30\n34.88%']
 ['False Infected\n20\n23.26%' 'True Uninfected\n13\n15.12%']]
```

In [63]:
```
import seaborn as sns
```

In [62]:
```
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(cf_matrix, columns=np.unique(y_true), index = np.unique(y
_true))
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, cmap="Blues", annot=labels ,annot_kws={"size": 16},fmt = ""
)
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x1e711d78188>



In [ ]: