In [2]:

```
# mount the google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:

```
# file path where the dataset resides
filedir = "/content/drive/MyDrive/croppart1"
```

In [4]:

```
import os
files=os.listdir(filedir)
```

In [ ]:

```
# Read each image using opencv , resize it to 48x48
# append the image on images list and  age number  in the age_list
import cv2
age_list=[]
images=[]
for file in files:
  age=int(file.split('_')[0])
  path=filedir+'/'+file
  image=cv2.imread(path)
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  image= cv2.resize(image,(48,48))
  images.append(image)
  age_list.append(age)

#saving images , age_list  by pickle
import pickle
with open('/content/drive/MyDrive/age/age_images.pkl', 'wb') as f:
  pickle.dump(images, f)
with open('/content/drive/MyDrive/age/age_list.pkl', 'wb') as f:
  pickle.dump(age_list, f)
```

In [25]:

```
import pickle
with open('/content/drive/MyDrive/age/age_images.pkl', 'rb') as f:
  images = pickle.load(f)


with open('/content/drive/MyDrive/age/age_list.pkl', 'rb') as f:
  age_list = pickle.load(f)


from google.colab.patches import cv2_imshow
cv2_imshow(images[24])
print(age_list[24])
import numpy as np
images1 = np.squeeze(images)
print(images1.shape)
images_f=np.array(images1)
ages_f=np.array(age_list)
np.save(filedir+'image.npy',images_f)
np.save(filedir+'age.npy',ages_f)


labels=[]
i=0
while i<len(age_list):
  label=age_list[i]
  labels.append(label)
  i+=1
#print(labels)
classes = []
# 0 : corresponds to kid
# 1 : corresponds to Adult
# 2 : corresponds to Middle Aged
# 3 : corresponds to Old Aged
for i in labels:
    i = int(i)
    if i <= 15:
        classes.append(0)
    if (i>15) and (i<=30):
        classes.append(1)
    if (i>30) and (i<60):
        classes.append(2)
    if i>=60:
        classes.append(3)
#print(classes)

from keras.utils.np_utils import to_categorical

images_f = images_f.astype('float32')
images_f_2=images_f/255
categorical_labels = to_categorical(classes, num_classes=4)

labels_f=np.array(classes)

images_f_2.shape

import tensorflow as tf
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test= train_test_split(images_f_2, categorical_labels,test_
size=0.15)
```

```
# print(Y_train)
# print(Y_test)
# model with 4 convolution layers with sigmoid as activation function
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten,BatchNormalization
from tensorflow.keras.layers import Dense, MaxPooling2D,Conv2D
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Flatten, InputLayer

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation=
'relu', input_shape=(48,48,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation=
'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(4, activation='sigmoid'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
29
(9780, 48, 48, 3)
```

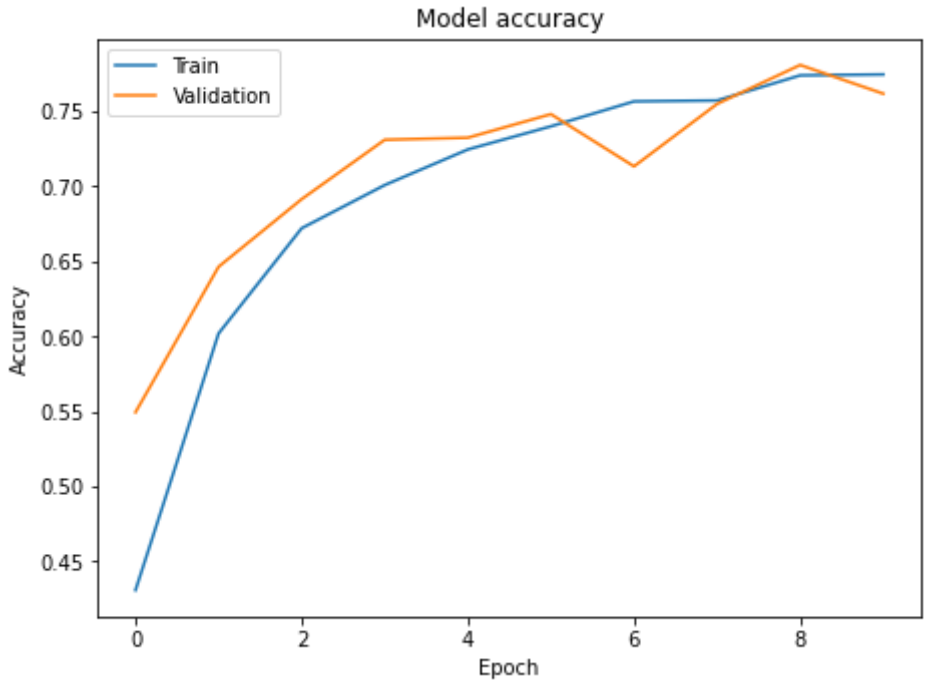
In [8]:

```
History = model.fit(X_train,
         Y_train,
         batch_size=64,
         epochs=10,
         validation_data=(X_test, Y_test))

model.save("/content/drive/MyDrive/age-model")
```

```
Epoch 1/10
130/130 [==============================] - 37s 274ms/step - loss: 1.3030 -
accuracy: 0.3946 - val_loss: 1.0821 - val_accuracy: 0.5494
Epoch 2/10
130/130 [==============================] - 35s 272ms/step - loss: 1.0017 -
accuracy: 0.5706 - val_loss: 0.8545 - val_accuracy: 0.6462
Epoch 3/10
130/130 [==============================] - 35s 272ms/step - loss: 0.8056 -
accuracy: 0.6686 - val_loss: 0.7406 - val_accuracy: 0.6912
Epoch 4/10
130/130 [==============================] - 35s 271ms/step - loss: 0.7288 -
accuracy: 0.6979 - val_loss: 0.6701 - val_accuracy: 0.7307
Epoch 5/10
130/130 [==============================] - 36s 277ms/step - loss: 0.6605 -
accuracy: 0.7220 - val_loss: 0.6325 - val_accuracy: 0.7321
Epoch 6/10
130/130 [==============================] - 38s 290ms/step - loss: 0.6510 -
accuracy: 0.7259 - val_loss: 0.6227 - val_accuracy: 0.7478
Epoch 7/10
130/130 [==============================] - 35s 272ms/step - loss: 0.5873 -
accuracy: 0.7561 - val_loss: 0.6669 - val_accuracy: 0.7130
Epoch 8/10
130/130 [==============================] - 35s 271ms/step - loss: 0.5713 -
accuracy: 0.7586 - val_loss: 0.5806 - val_accuracy: 0.7546
Epoch 9/10
130/130 [==============================] - 35s 270ms/step - loss: 0.5507 -
accuracy: 0.7667 - val_loss: 0.5646 - val_accuracy: 0.7805
Epoch 10/10
130/130 [==============================] - 35s 270ms/step - loss: 0.5369 -
accuracy: 0.7775 - val_loss: 0.5529 - val_accuracy: 0.7614
INFO:tensorflow:Assets written to: /content/drive/MyDrive/age-model/assets
```

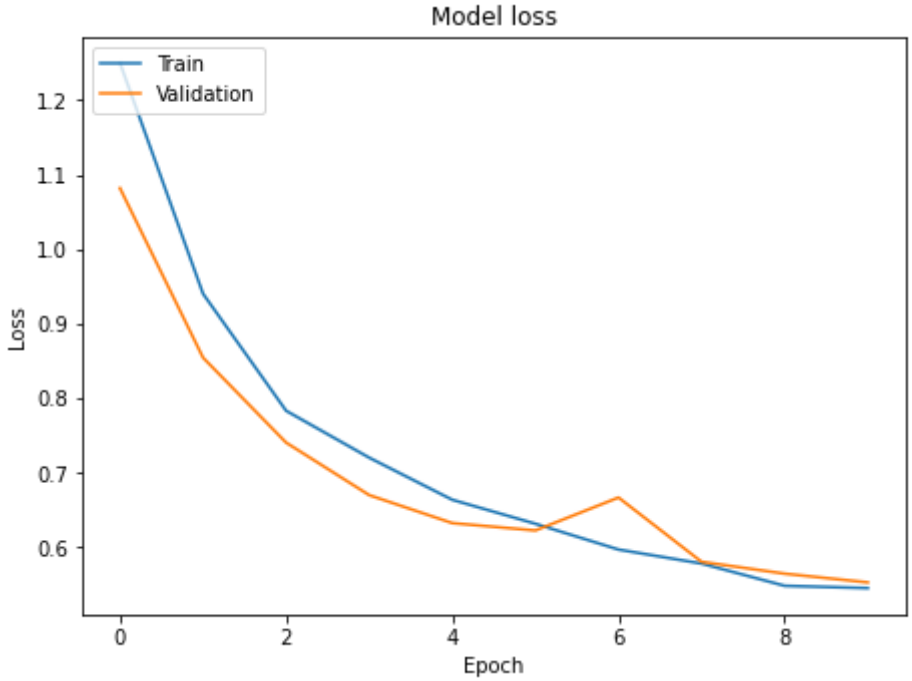
In [10]:

```
import matplotlib.pyplot as plt
plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [12]:

```
Pred=model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [13]:

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

i=0
Y_test_l=[]
Pred_l=[]
while(i<len(Pred)):
  Y_test_l.append(int(np.argmax(Y_test[i])))
  Pred_l.append(int(np.argmax(Pred[i])))
  i+=1
report=classification_report(Y_test_l, Pred_l)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90       552
           1       0.64      0.74      0.68       328
           2       0.66      0.66      0.66       371
           3       0.77      0.65      0.70       216

    accuracy                           0.76      1467
   macro avg       0.75      0.73      0.74      1467
weighted avg       0.77      0.76      0.76      1467
```

In [14]:

```
results=confusion_matrix(Y_test_l,Pred_l)
import seaborn as sns
sns.heatmap(results,annot=True)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fba2ae76f50>
```

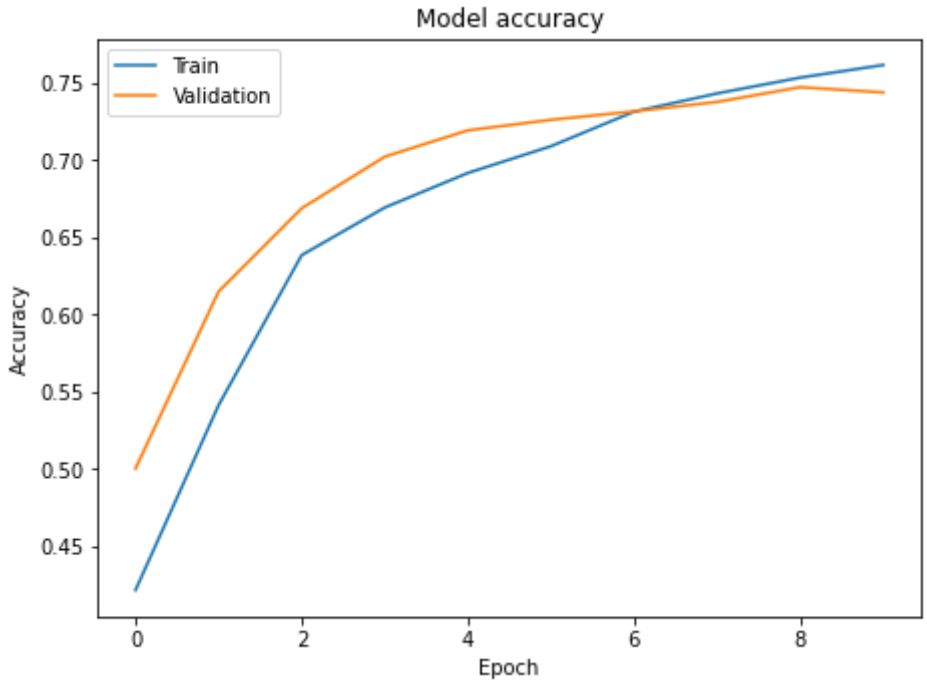
In [15]:

```
# model with 4 convolution layers with softmax as activation function
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten,BatchNormalization
from tensorflow.keras.layers import Dense, MaxPooling2D,Conv2D
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Flatten, InputLayer

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation=
'relu', input_shape=(48,48,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation=
'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(4, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
History = model.fit(X_train,
          Y_train,
          batch_size=64,
          epochs=10,
          validation_data=(X_test, Y_test))

model.save("/content/drive/MyDrive/age-model")
```

```
Epoch 1/10
130/130 [==============================] - 36s 270ms/step - loss: 1.3102 -
accuracy: 0.3789 - val_loss: 1.1507 - val_accuracy: 0.5003
Epoch 2/10
130/130 [==============================] - 35s 270ms/step - loss: 1.1068 -
accuracy: 0.5193 - val_loss: 0.9623 - val_accuracy: 0.6149
Epoch 3/10
130/130 [==============================] - 35s 271ms/step - loss: 0.9056 -
accuracy: 0.6219 - val_loss: 0.7905 - val_accuracy: 0.6687
Epoch 4/10
130/130 [==============================] - 35s 273ms/step - loss: 0.7920 -
accuracy: 0.6658 - val_loss: 0.7401 - val_accuracy: 0.7021
Epoch 5/10
130/130 [==============================] - 35s 271ms/step - loss: 0.7384 -
accuracy: 0.6929 - val_loss: 0.6914 - val_accuracy: 0.7192
Epoch 6/10
130/130 [==============================] - 35s 271ms/step - loss: 0.6925 -
accuracy: 0.7086 - val_loss: 0.6704 - val_accuracy: 0.7260
Epoch 7/10
130/130 [==============================] - 35s 273ms/step - loss: 0.6414 -
accuracy: 0.7360 - val_loss: 0.6437 - val_accuracy: 0.7314
Epoch 8/10
130/130 [==============================] - 35s 273ms/step - loss: 0.6181 -
accuracy: 0.7458 - val_loss: 0.6340 - val_accuracy: 0.7376
Epoch 9/10
130/130 [==============================] - 35s 270ms/step - loss: 0.5914 -
accuracy: 0.7581 - val_loss: 0.6156 - val_accuracy: 0.7471
Epoch 10/10
130/130 [==============================] - 35s 272ms/step - loss: 0.5801 -
accuracy: 0.7577 - val_loss: 0.6080 - val_accuracy: 0.7437
INFO:tensorflow:Assets written to: /content/drive/MyDrive/age-model/assets
```

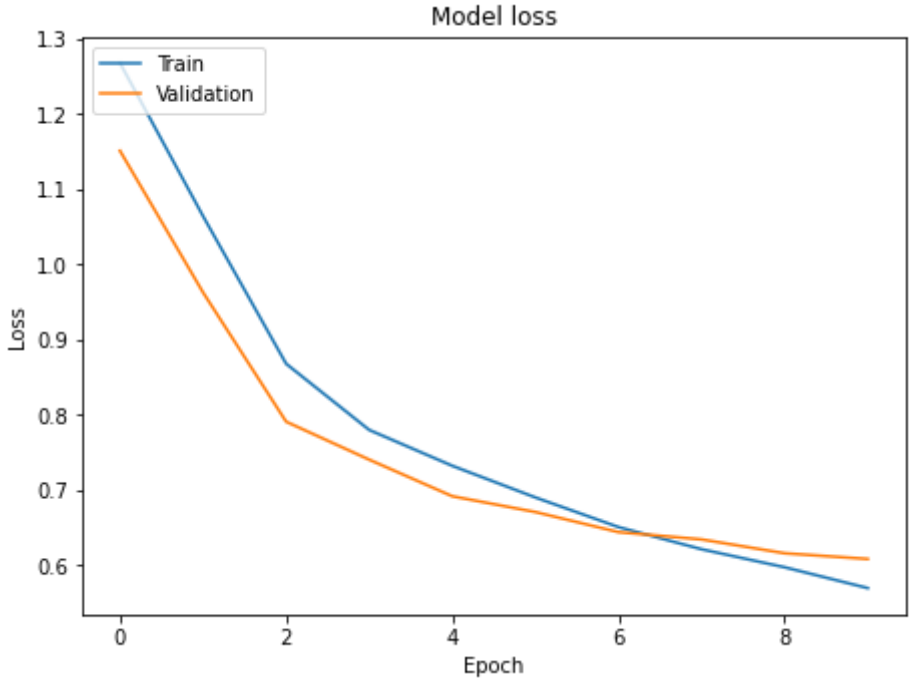
In [16]:

```
import matplotlib.pyplot as plt
plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [17]:

```
Pred=model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [18]:

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

i=0
Y_test_l=[]
Pred_l=[]
while(i<len(Pred)):
  Y_test_l.append(int(np.argmax(Y_test[i])))
  Pred_l.append(int(np.argmax(Pred[i])))
  i+=1
report=classification_report(Y_test_l, Pred_l)
print(report)
```

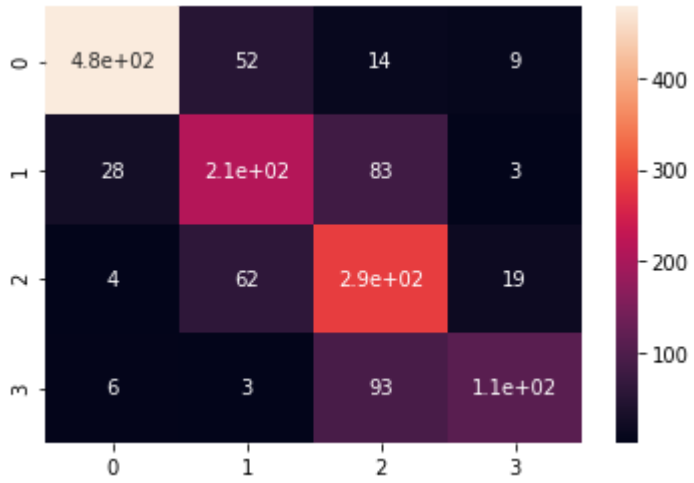
```
              precision    recall  f1-score   support

           0       0.93      0.86      0.89       552
           1       0.65      0.65      0.65       328
           2       0.60      0.77      0.68       371
           3       0.79      0.53      0.63       216

    accuracy                           0.74      1467
   macro avg       0.74      0.70      0.71      1467
weighted avg       0.76      0.74      0.75      1467
```

In [19]:

```
results=confusion_matrix(Y_test_l,Pred_l)
import seaborn as sns
sns.heatmap(results,annot=True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fba2037a3d0>
```



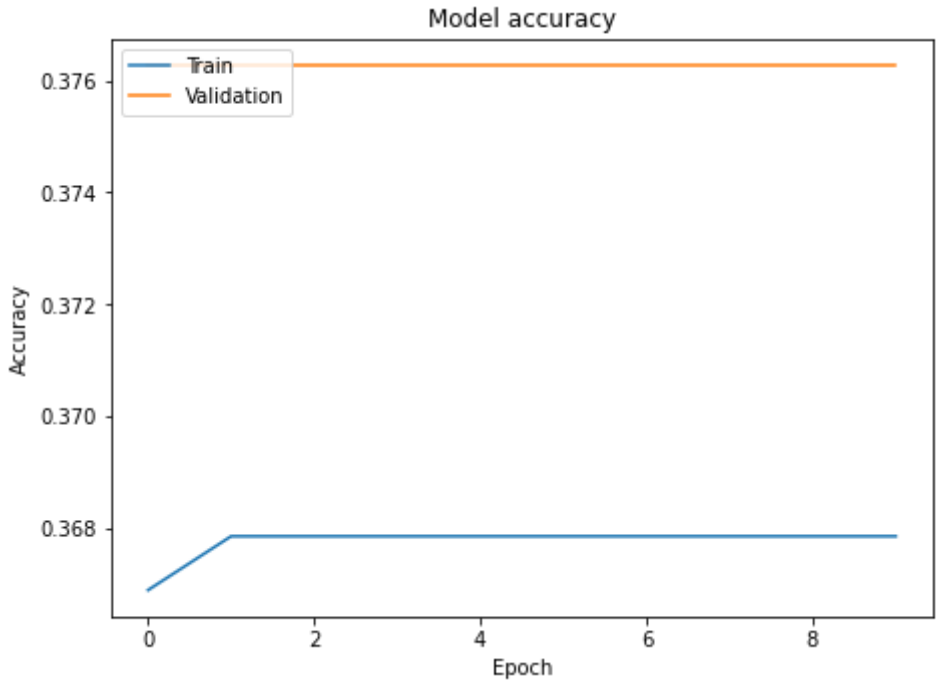
In [20]:

```
# model with 4 convolution layers with relu as activation function
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten,BatchNormalization
from tensorflow.keras.layers import Dense, MaxPooling2D,Conv2D
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Flatten, InputLayer

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation=
'relu', input_shape=(48,48,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation=
'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=2, padding='same', activation
='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(4, activation='relu'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
History = model.fit(X_train,
          Y_train,
          batch_size=64,
          epochs=10,
          validation_data=(X_test, Y_test))

model.save("/content/drive/MyDrive/age-model")
```

```
Epoch 1/10
130/130 [==============================] - 37s 277ms/step - loss: nan - ac
curacy: 0.3598 - val_loss: nan - val_accuracy: 0.3763
Epoch 2/10
130/130 [==============================] - 36s 277ms/step - loss: nan - ac
curacy: 0.3688 - val_loss: nan - val_accuracy: 0.3763
Epoch 3/10
130/130 [==============================] - 36s 275ms/step - loss: nan - ac
curacy: 0.3635 - val_loss: nan - val_accuracy: 0.3763
Epoch 4/10
130/130 [==============================] - 36s 274ms/step - loss: nan - ac
curacy: 0.3628 - val_loss: nan - val_accuracy: 0.3763
Epoch 5/10
130/130 [==============================] - 36s 274ms/step - loss: nan - ac
curacy: 0.3709 - val_loss: nan - val_accuracy: 0.3763
Epoch 6/10
130/130 [==============================] - 35s 272ms/step - loss: nan - ac
curacy: 0.3677 - val_loss: nan - val_accuracy: 0.3763
Epoch 7/10
130/130 [==============================] - 35s 271ms/step - loss: nan - ac
curacy: 0.3564 - val_loss: nan - val_accuracy: 0.3763
Epoch 8/10
130/130 [==============================] - 35s 273ms/step - loss: nan - ac
curacy: 0.3681 - val_loss: nan - val_accuracy: 0.3763
Epoch 9/10
130/130 [==============================] - 35s 270ms/step - loss: nan - ac
curacy: 0.3669 - val_loss: nan - val_accuracy: 0.3763
Epoch 10/10
130/130 [==============================] - 35s 273ms/step - loss: nan - ac
curacy: 0.3667 - val_loss: nan - val_accuracy: 0.3763
INFO:tensorflow:Assets written to: /content/drive/MyDrive/age-model/assets
```

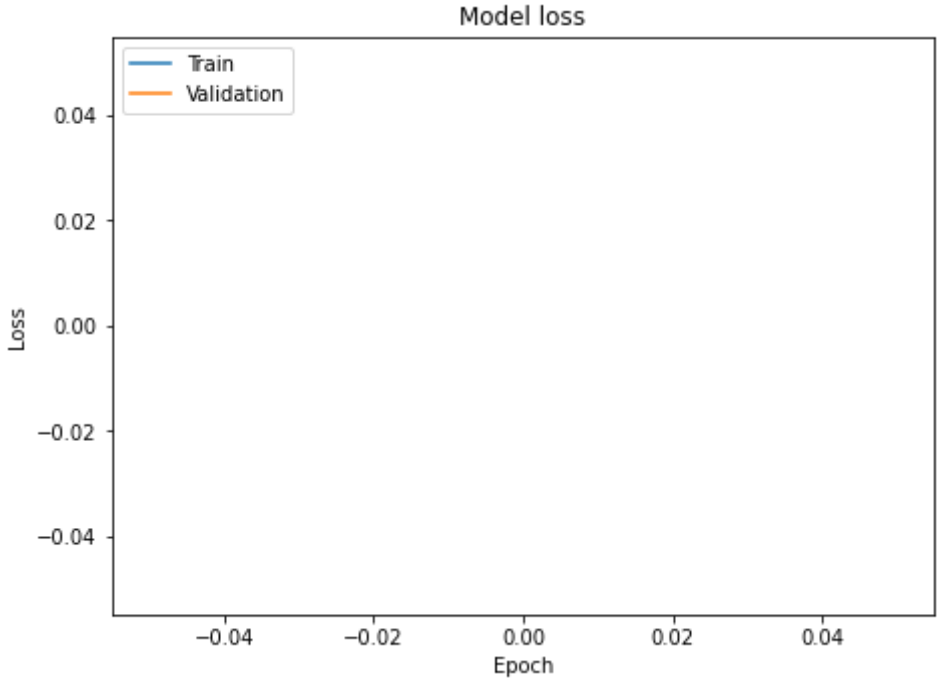
In [21]:

```
import matplotlib.pyplot as plt
plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [22]:

```
Pred=model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                    wspace=0.35)
```



In [23]:

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

i=0
Y_test_l=[]
Pred_l=[]
while(i<len(Pred)):
  Y_test_l.append(int(np.argmax(Y_test[i])))
  Pred_l.append(int(np.argmax(Pred[i])))
  i+=1
report=classification_report(Y_test_l, Pred_l)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.38      1.00      0.55       552
           1       0.00      0.00      0.00       328
           2       0.00      0.00      0.00       371
           3       0.00      0.00      0.00       216

    accuracy                           0.38      1467
   macro avg       0.09      0.25      0.14      1467
weighted avg       0.14      0.38      0.21      1467
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:
1272: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` pa
rameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [24]:

```
results=confusion_matrix(Y_test_l,Pred_l)
import seaborn as sns
sns.heatmap(results,annot=True)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fba202c7a50>
```