In [51]:

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
all drive.mount("/content/drive", force_remount=True).
```

In [52]:

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

In [53]:

```
# print the name of the folders present in the dataset
import os
files=os.listdir(filedir)
print(files)
```

```
['happy', 'disgust', 'contempt', 'sadness', 'fear', 'surprise', 'anger']
```

In [54]:

```
# storing the emotions in the list
emotion =['happy', 'disgust', 'contempt', 'sadness', 'fear', 'surprise', 'anger']
```

In [ ]:

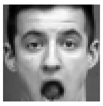
```
# Read each image using opencv , resize it to 48x48
# append the image on images list and  label in the labels list
import cv2
from google.colab.patches import cv2_imshow
i=0
images=[]
labels=[]
for file in files:
  idx=emotion.index(file)
  label=idx
  full_path=filedir+'/'+file
  files_exp= os.listdir(full_path)
  counter = 0

  for file_2 in files_exp:
    file_main=full_path+'/'+file_2
    image= cv2.imread(file_main)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image= cv2.resize(image,(48,48))
    images.append(image)
    labels.append(label)
    i+=1

#saving images , labels  by pickle
import pickle
with open('/content/drive/MyDrive/emotion/emotion_images.pkl', 'wb') as f:
  pickle.dump(images, f)
with open('/content/drive/MyDrive/emotion/emotion_labels.pkl', 'wb') as f:
  pickle.dump(labels, f)
```

In [55]:

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


with open('/content/drive/MyDrive/emotion/emotion_labels.pkl', 'rb') as f:
  labels = pickle.load(f)
```

In [56]:

```
#showing some sample images of the dataset
cv2_imshow(images[200])
```



In [57]:

```
#importing tensor flow libraries
import tensorflow as tf
from sklearn.model_selection import train_test_split
```

In [58]:

```
#data preprocessing
#converting images and labels into numpy arrays with numpy
#images are normalized by dividing it with 255
import numpy as np
images_f=np.array(images)
labels_f=np.array(labels)

images_f_2=images_f/255
```

In [59]:

```
images_f_2.shape
```

Out[59]:

```
(981, 48, 48, 3)
```

In [60]:

```
# corresponding to 7 emotions there are 7 classes
num_of_classes=7
labels_encoded=tf.keras.utils.to_categorical(labels_f,num_classes=num_of_classes)
```

In [61]:

```
#splitting the dataset into training and test data set
X_train, X_test, Y_train, Y_test= train_test_split(images_f_2, labels_encoded,test_size
=0.25)
```

In [63]:

```
# Number of convolutional Layers = 4 and activation function as sigmoid
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
from tensorflow.keras.callbacks import ModelCheckpoint

def Convolution(input_tensor,filters):

    x = Conv2D(filters=filters,kernel_size=(3, 3),padding = 'same',strides=(1, 1),kerne
l_regularizer=l2(0.001))(input_tensor)
    x = Dropout(0.1)(x)
    x= Activation('relu')(x)

    return x
def model(input_shape):
  inputs = Input((input_shape))

  convolution_1= Convolution(inputs,32)
  maxpooling_1 = MaxPooling2D(pool_size = (2,2)) (convolution_1)
  convolution_2 = Convolution(maxpooling_1,64)
  maxpooling_2 = MaxPooling2D(pool_size = (2, 2)) (convolution_2)
  convolution_3 = Convolution(maxpooling_2,128)
  maxpooling_3 = MaxPooling2D(pool_size = (2, 2)) (convolution_3)
  convolution_4 = Convolution(maxpooling_2,256)
  maxpooling_4 = MaxPooling2D(pool_size = (2, 2)) (convolution_4)
  flatten= Flatten() (maxpooling_4)
  dense_1= Dense(256,activation='relu')(flatten)
  drop_1=Dropout(0.2)(dense_1)
  output= Dense(7,activation='sigmoid')(drop_1)

  model = Model(inputs=[inputs], outputs=[output])

  model.compile(loss="categorical_crossentropy", optimizer="Adam",
        metrics=["accuracy"])
  return model
```

In [64]:

```
Model=model(input_shape = (48,48,3))
Model.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_8 (InputLayer)         [(None, 48, 48, 3)]       0         
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 48, 48, 32)        896       
_________________________________________________________________
dropout_35 (Dropout)         (None, 48, 48, 32)        0         
_________________________________________________________________
activation_28 (Activation)   (None, 48, 48, 32)        0         
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 24, 24, 32)        0         
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 24, 24, 64)        18496     
_________________________________________________________________
dropout_36 (Dropout)         (None, 24, 24, 64)        0         
_________________________________________________________________
activation_29 (Activation)   (None, 24, 24, 64)        0         
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 12, 12, 64)        0         
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 12, 12, 256)       147712    
_________________________________________________________________
dropout_38 (Dropout)         (None, 12, 12, 256)       0         
_________________________________________________________________
activation_31 (Activation)   (None, 12, 12, 256)       0         
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 6, 6, 256)         0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 9216)              0         
_________________________________________________________________
dense_14 (Dense)             (None, 256)               2359552   
_________________________________________________________________
dropout_39 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_15 (Dense)             (None, 7)                 1799      
=================================================================
Total params: 2,528,455
Trainable params: 2,528,455
Non-trainable params: 0
_________________________________________________________________
```

In [65]:

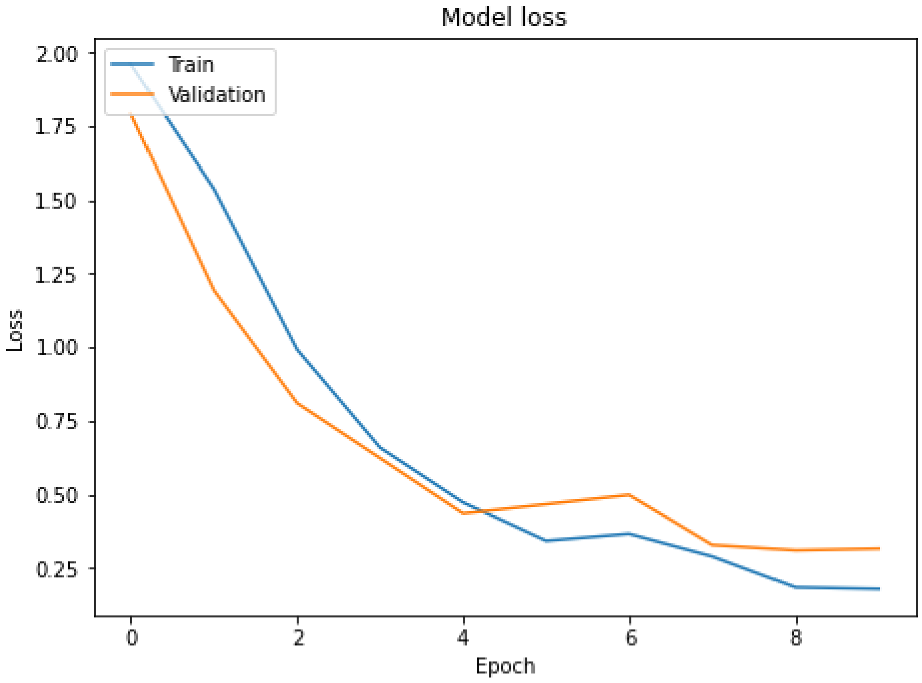
```
from tensorflow.keras.callbacks import ModelCheckpoint


fle_s='Emotion_detection_2.h5'
checkpointer = ModelCheckpoint(fle_s, monitor='loss',verbose=1,save_best_only=True,save
_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]

History=Model.fit(X_train,Y_train,batch_size=32,validation_data=(X_test,Y_test),epochs=
10,callbacks=[callback_list])
score = Model.evaluate(X_train, Y_train)
score = Model.evaluate(X_test, Y_test)
```

```
Epoch 1/10
23/23 [==============================] - 8s 299ms/step - loss: 2.0487 - ac
curacy: 0.2005 - val_loss: 1.7922 - val_accuracy: 0.3008

Epoch 00001: loss improved from inf to 1.96073, saving model to Emotion_de
tection_2.h5
Epoch 2/10
23/23 [==============================] - 6s 277ms/step - loss: 1.6675 - ac
curacy: 0.3743 - val_loss: 1.1922 - val_accuracy: 0.6789

Epoch 00002: loss improved from 1.96073 to 1.53566, saving model to Emotio
n_detection_2.h5
Epoch 3/10
23/23 [==============================] - 6s 278ms/step - loss: 1.0722 - ac
curacy: 0.6484 - val_loss: 0.8096 - val_accuracy: 0.7561

Epoch 00003: loss improved from 1.53566 to 0.99184, saving model to Emotio
n_detection_2.h5
Epoch 4/10
23/23 [==============================] - 6s 279ms/step - loss: 0.6512 - ac
curacy: 0.8149 - val_loss: 0.6223 - val_accuracy: 0.8333

Epoch 00004: loss improved from 0.99184 to 0.65803, saving model to Emotio
n_detection_2.h5
Epoch 5/10
23/23 [==============================] - 6s 280ms/step - loss: 0.5180 - ac
curacy: 0.8646 - val_loss: 0.4345 - val_accuracy: 0.8984

Epoch 00005: loss improved from 0.65803 to 0.47198, saving model to Emotio
n_detection_2.h5
Epoch 6/10
23/23 [==============================] - 6s 279ms/step - loss: 0.3371 - ac
curacy: 0.9152 - val_loss: 0.4661 - val_accuracy: 0.8415

Epoch 00006: loss improved from 0.47198 to 0.34171, saving model to Emotio
n_detection_2.h5
Epoch 7/10
23/23 [==============================] - 6s 280ms/step - loss: 0.3756 - ac
curacy: 0.8924 - val_loss: 0.4977 - val_accuracy: 0.8659

Epoch 00007: loss did not improve from 0.34171
Epoch 8/10
23/23 [==============================] - 6s 279ms/step - loss: 0.3046 - ac
curacy: 0.9245 - val_loss: 0.3278 - val_accuracy: 0.9390

Epoch 00008: loss improved from 0.34171 to 0.28902, saving model to Emotio
n_detection_2.h5
Epoch 9/10
23/23 [==============================] - 6s 280ms/step - loss: 0.1913 - ac
curacy: 0.9711 - val_loss: 0.3098 - val_accuracy: 0.9350

Epoch 00009: loss improved from 0.28902 to 0.18413, saving model to Emotio
n_detection_2.h5
Epoch 10/10
23/23 [==============================] - 6s 279ms/step - loss: 0.1869 - ac
curacy: 0.9676 - val_loss: 0.3154 - val_accuracy: 0.9431

Epoch 00010: loss improved from 0.18413 to 0.17886, saving model to Emotio
n_detection_2.h5
23/23 [==============================] - 1s 61ms/step - loss: 0.1531 - acc
uracy: 0.9878
```

```
8/8 [==============================] - 0s 58ms/step - loss: 0.3154 - accur
acy: 0.9431
```

In [66]:

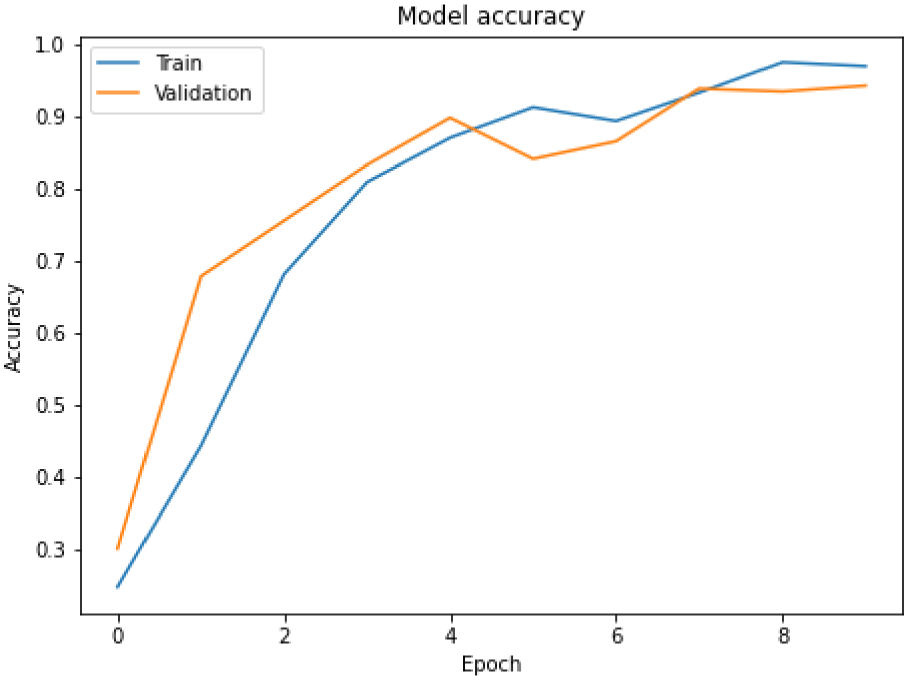
```
Pred=Model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0,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



In [67]:

```
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)

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       0.88      1.00      0.93        56
           1       0.97      1.00      0.99        37
           2       1.00      0.92      0.96        12
           3       1.00      0.86      0.92        21
           4       1.00      0.65      0.79        17
           5       1.00      0.96      0.98        74
           6       0.85      0.97      0.90        29

    accuracy                           0.94       246
   macro avg       0.96      0.91      0.92       246
weighted avg       0.95      0.94      0.94       246
```



In [68]:

```
# changing activation function to relu


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
from tensorflow.keras.callbacks import ModelCheckpoint

def Convolution(input_tensor,filters):

    x = Conv2D(filters=filters,kernel_size=(3, 3),padding = 'same',strides=(1, 1),kernel_regularizer=l2(0.001))(input_tensor)
    x = Dropout(0.1)(x)
    x= Activation('relu')(x)

    return x
def model(input_shape):
  inputs = Input((input_shape))

  convolution_1= Convolution(inputs,32)
  maxpooling_1 = MaxPooling2D(pool_size = (2,2)) (convolution_1)
  convolution_2 = Convolution(maxpooling_1,64)
  maxpooling_2 = MaxPooling2D(pool_size = (2, 2)) (convolution_2)
  convolution_3 = Convolution(maxpooling_2,128)
  maxpooling_3 = MaxPooling2D(pool_size = (2, 2)) (convolution_3)
  convolution_4 = Convolution(maxpooling_2,256)
  maxpooling_4 = MaxPooling2D(pool_size = (2, 2)) (convolution_4)
  flatten= Flatten() (maxpooling_4)
  dense_1= Dense(256,activation='relu')(flatten)
  drop_1=Dropout(0.2)(dense_1)
  output= Dense(7,activation='relu')(drop_1)

  model = Model(inputs=[inputs], outputs=[output])

  model.compile(loss="categorical_crossentropy", optimizer="Adam",
        metrics=["accuracy"])
  return model
Model=model(input_shape = (48,48,3))
Model.summary()
from tensorflow.keras.callbacks import ModelCheckpoint


fle_s='Emotion_detection_2.h5'
checkpointer = ModelCheckpoint(fle_s, monitor='loss',verbose=1,save_best_only=True,save_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]

History=Model.fit(X_train,Y_train,batch_size=32,validation_data=(X_test,Y_test),epochs=10,callbacks=[callback_list])
score = Model.evaluate(X_train, Y_train)
score = Model.evaluate(X_test, Y_test)
```

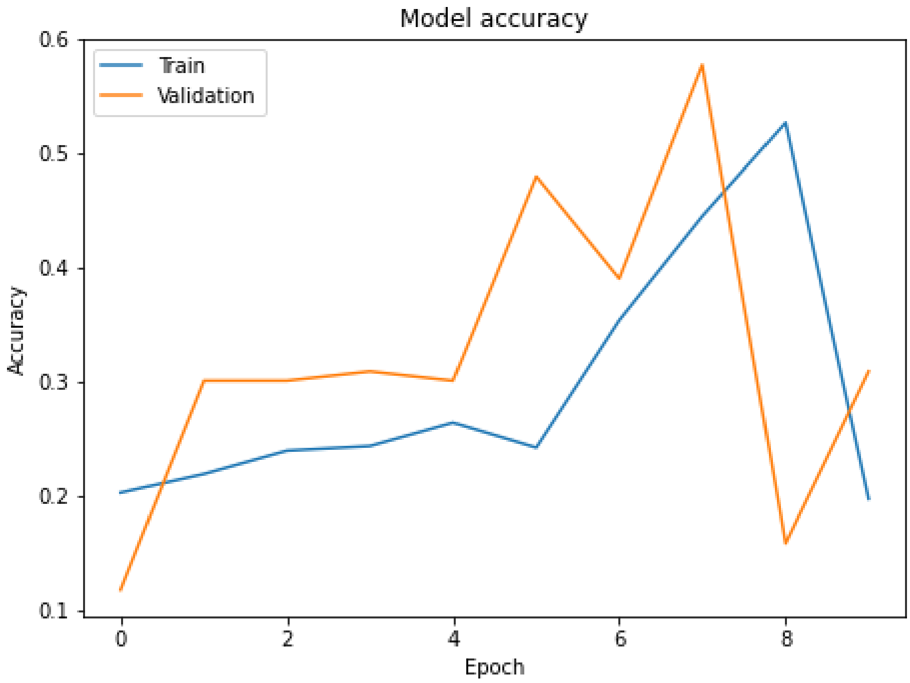
```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 48, 48, 3)]       0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 48, 48, 32)        896
_________________________________________________________________
dropout_40 (Dropout)         (None, 48, 48, 32)        0
_________________________________________________________________
activation_32 (Activation)   (None, 48, 48, 32)        0
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
dropout_41 (Dropout)         (None, 24, 24, 64)        0
_________________________________________________________________
activation_33 (Activation)   (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 12, 12, 256)       147712
_________________________________________________________________
dropout_43 (Dropout)         (None, 12, 12, 256)       0
_________________________________________________________________
activation_35 (Activation)   (None, 12, 12, 256)       0
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               2359552
_________________________________________________________________
dropout_44 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_17 (Dense)             (None, 7)                 1799
=================================================================
Total params: 2,528,455
Trainable params: 2,528,455
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
23/23 [==============================] - 7s 294ms/step - loss: 5.1188 - ac
curacy: 0.2280 - val_loss: 3.0438 - val_accuracy: 0.1179

Epoch 00001: loss improved from inf to 4.06006, saving model to Emotion_de
tection_2.h5
Epoch 2/10
23/23 [==============================] - 7s 288ms/step - loss: 3.0283 - ac
curacy: 0.1896 - val_loss: 2.9949 - val_accuracy: 0.3008

Epoch 00002: loss improved from 4.06006 to 3.05544, saving model to Emotio
n_detection_2.h5
Epoch 3/10
23/23 [==============================] - 7s 287ms/step - loss: 3.1532 - ac
curacy: 0.2408 - val_loss: 2.9873 - val_accuracy: 0.3008

Epoch 00003: loss improved from 3.05544 to 3.03658, saving model to Emotio
n_detection_2.h5
Epoch 4/10
```

```
23/23 [==============================] - 7s 284ms/step - loss: 3.0844 - ac
curacy: 0.2466 - val_loss: 2.9778 - val_accuracy: 0.3089

Epoch 00004: loss did not improve from 3.03658
Epoch 5/10
23/23 [==============================] - 7s 284ms/step - loss: 3.0551 - ac
curacy: 0.2648 - val_loss: 3.2961 - val_accuracy: 0.3008

Epoch 00005: loss improved from 3.03658 to 3.01834, saving model to Emotio
n_detection_2.h5
Epoch 6/10
23/23 [==============================] - 7s 284ms/step - loss: 3.0386 - ac
curacy: 0.2519 - val_loss: 2.9296 - val_accuracy: 0.4797

Epoch 00006: loss did not improve from 3.01834
Epoch 7/10
23/23 [==============================] - 6s 282ms/step - loss: 2.6745 - ac
curacy: 0.3655 - val_loss: 1.6339 - val_accuracy: 0.3902

Epoch 00007: loss improved from 3.01834 to 2.54982, saving model to Emotio
n_detection_2.h5
Epoch 8/10
23/23 [==============================] - 7s 286ms/step - loss: 1.7317 - ac
curacy: 0.3763 - val_loss: 1.4207 - val_accuracy: 0.5772

Epoch 00008: loss improved from 2.54982 to 1.80579, saving model to Emotio
n_detection_2.h5
Epoch 9/10
23/23 [==============================] - 7s 283ms/step - loss: 1.5502 - ac
curacy: 0.5804 - val_loss: 1.8511 - val_accuracy: 0.1585

Epoch 00009: loss improved from 1.80579 to 1.68127, saving model to Emotio
n_detection_2.h5
Epoch 10/10
23/23 [==============================] - 6s 281ms/step - loss: 1.9971 - ac
curacy: 0.1427 - val_loss: 1.7893 - val_accuracy: 0.3089

Epoch 00010: loss did not improve from 1.68127
23/23 [==============================] - 1s 62ms/step - loss: 1.8440 - acc
uracy: 0.2476
8/8 [==============================] - 1s 61ms/step - loss: 1.7893 - accur
acy: 0.3089
```

In [69]:

```
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)

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       0.88      1.00      0.93        56
           1       0.97      1.00      0.99        37
           2       1.00      0.92      0.96        12
           3       1.00      0.86      0.92        21
           4       1.00      0.65      0.79        17
           5       1.00      0.96      0.98        74
           6       0.85      0.97      0.90        29

    accuracy                           0.94       246
   macro avg       0.96      0.91      0.92       246
weighted avg       0.95      0.94      0.94       246
```



In [70]:

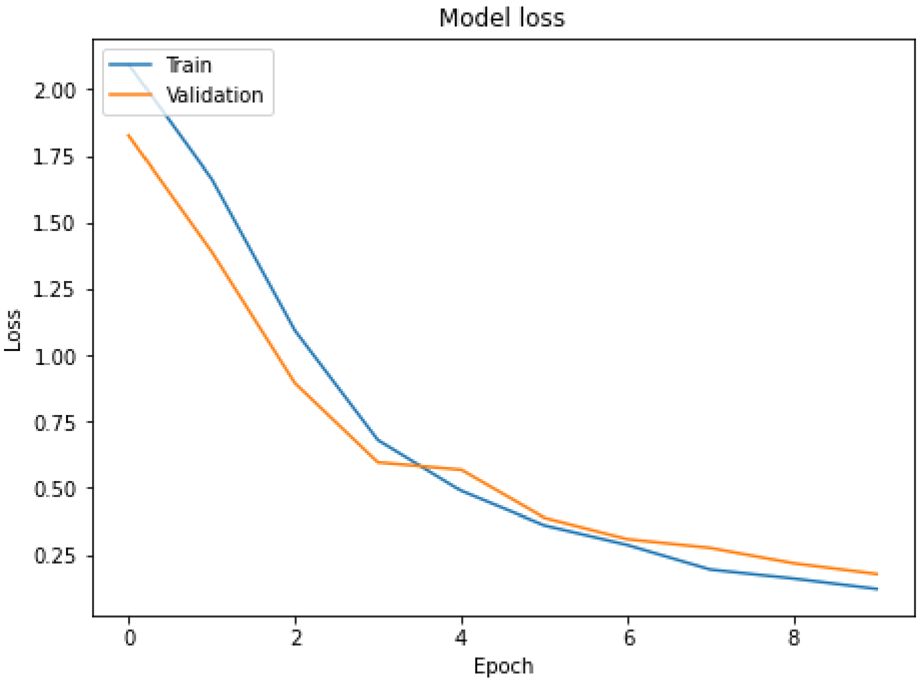
```
# changing activation function to softmax
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten,BatchNormalization
from tensorflow.keras.layers import Dense, MaxPooling2D,Conv2D
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint

def Convolution(input_tensor,filters):

    x = Conv2D(filters=filters,kernel_size=(3, 3),padding = 'same',strides=(1, 1),kerne
l_regularizer=l2(0.001))(input_tensor)
    x = Dropout(0.1)(x)
    x= Activation('relu')(x)

    return x
def model(input_shape):
  inputs = Input((input_shape))

  conv_1= Convolution(inputs,32)
  maxp_1 = MaxPooling2D(pool_size = (2,2)) (conv_1)
  conv_2 = Convolution(maxp_1,64)
  maxp_2 = MaxPooling2D(pool_size = (2, 2)) (conv_2)
  conv_3 = Convolution(maxp_1,128)
  maxp_3 = MaxPooling2D(pool_size = (3, 3)) (conv_3)
  conv_4 = Convolution(maxp_1,256)
  maxp_4 = MaxPooling2D(pool_size = (3, 3)) (conv_4)
  flatten= Flatten() (maxp_4)
  dense_1= Dense(256,activation='relu')(flatten)
  drop_1=Dropout(0.2)(dense_1)
  output= Dense(7,activation='softmax')(drop_1)

  model = Model(inputs=[inputs], outputs=[output])

  model.compile(loss="categorical_crossentropy", optimizer="Adam",
        metrics=["accuracy"])
  return model
Model=model(input_shape = (48,48,3))
Model.summary()



History=Model.fit(X_train,Y_train,batch_size=32,validation_data=(X_test,Y_test),epochs=
10,callbacks=[callback_list])
score = Model.evaluate(X_train, Y_train)
score = Model.evaluate(X_test, Y_test)
```

```
Model: "model_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_10 (InputLayer)        [(None, 48, 48, 3)]       0         
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 48, 48, 32)        896       
_________________________________________________________________
dropout_45 (Dropout)         (None, 48, 48, 32)        0         
_________________________________________________________________
activation_36 (Activation)   (None, 48, 48, 32)        0         
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 24, 24, 32)        0         
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 24, 24, 256)       73984     
_________________________________________________________________
dropout_48 (Dropout)         (None, 24, 24, 256)       0         
_________________________________________________________________
activation_39 (Activation)   (None, 24, 24, 256)       0         
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 8, 8, 256)         0         
_________________________________________________________________
flatten_9 (Flatten)          (None, 16384)             0         
_________________________________________________________________
dense_18 (Dense)             (None, 256)               4194560   
_________________________________________________________________
dropout_49 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_19 (Dense)             (None, 7)                 1799      
=================================================================
Total params: 4,271,239
Trainable params: 4,271,239
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
23/23 [==============================] - 9s 361ms/step - loss: 2.4079 - ac
curacy: 0.1952 - val_loss: 1.8262 - val_accuracy: 0.5081

Epoch 00001: loss did not improve from 1.68127
Epoch 2/10
23/23 [==============================] - 8s 349ms/step - loss: 1.7364 - ac
curacy: 0.4118 - val_loss: 1.3885 - val_accuracy: 0.6463

Epoch 00002: loss improved from 1.68127 to 1.66208, saving model to Emotio
n_detection_2.h5
Epoch 3/10
23/23 [==============================] - 8s 353ms/step - loss: 1.1704 - ac
curacy: 0.6195 - val_loss: 0.8949 - val_accuracy: 0.7317

Epoch 00003: loss improved from 1.66208 to 1.09324, saving model to Emotio
n_detection_2.h5
Epoch 4/10
23/23 [==============================] - 8s 352ms/step - loss: 0.7086 - ac
curacy: 0.7537 - val_loss: 0.5967 - val_accuracy: 0.8415

Epoch 00004: loss improved from 1.09324 to 0.68071, saving model to Emotio
n_detection_2.h5
Epoch 5/10
23/23 [==============================] - 8s 355ms/step - loss: 0.5508 - ac
curacy: 0.8362 - val_loss: 0.5688 - val_accuracy: 0.8374
```

```
Epoch 00005: loss improved from 0.68071 to 0.48994, saving model to Emotio
n_detection_2.h5
Epoch 6/10
23/23 [==============================] - 8s 350ms/step - loss: 0.3917 - ac
curacy: 0.8793 - val_loss: 0.3889 - val_accuracy: 0.9106

Epoch 00006: loss improved from 0.48994 to 0.36063, saving model to Emotio
n_detection_2.h5
Epoch 7/10
23/23 [==============================] - 8s 353ms/step - loss: 0.2909 - ac
curacy: 0.9189 - val_loss: 0.3089 - val_accuracy: 0.9553

Epoch 00007: loss improved from 0.36063 to 0.28712, saving model to Emotio
n_detection_2.h5
Epoch 8/10
23/23 [==============================] - 8s 352ms/step - loss: 0.2070 - ac
curacy: 0.9559 - val_loss: 0.2762 - val_accuracy: 0.9472

Epoch 00008: loss improved from 0.28712 to 0.19487, saving model to Emotio
n_detection_2.h5
Epoch 9/10
23/23 [==============================] - 8s 354ms/step - loss: 0.1674 - ac
curacy: 0.9612 - val_loss: 0.2184 - val_accuracy: 0.9715

Epoch 00009: loss improved from 0.19487 to 0.16079, saving model to Emotio
n_detection_2.h5
Epoch 10/10
23/23 [==============================] - 8s 354ms/step - loss: 0.1291 - ac
curacy: 0.9829 - val_loss: 0.1781 - val_accuracy: 0.9797

Epoch 00010: loss improved from 0.16079 to 0.12176, saving model to Emotio
n_detection_2.h5
23/23 [==============================] - 2s 77ms/step - loss: 0.1085 - acc
uracy: 0.9986
8/8 [==============================] - 1s 72ms/step - loss: 0.1781 - accur
acy: 0.9797
```

In [71]:

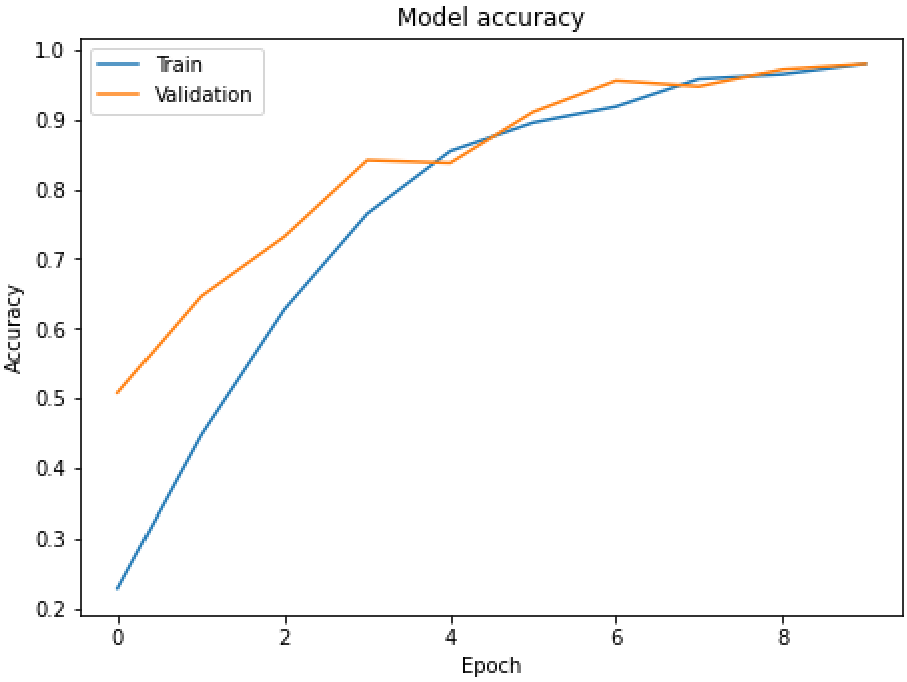
```
Pred=Model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0,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



In [72]:

```
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)

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       0.95      1.00      0.97        56
           1       1.00      1.00      1.00        37
           2       1.00      1.00      1.00        12
           3       1.00      0.90      0.95        21
           4       1.00      1.00      1.00        17
           5       0.97      0.96      0.97        74
           6       1.00      1.00      1.00        29

    accuracy                           0.98       246
   macro avg       0.99      0.98      0.98       246
weighted avg       0.98      0.98      0.98       246
```



In [ ]:

```
test_image(72,images_f,images_f_2,Model)
```

```
Label actual:  happy
Predicted Label: happy
```

In [ ]:

```
test_image(132,images_f,images_f_2,Model)
```

```
Label actual:  happy
Predicted Label: happy
```

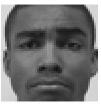
In [ ]:

```
test_image(147,images_f,images_f_2,Model)
```

```
Label actual:  happy
Predicted Label: happy
```

In [ ]:

```
test_image(500,images_f,images_f_2,Model)
```



```
Label actual:  sadness
Predicted Label: sadness
```

In [ ]:

```
test_image(300,images_f,images_f_2,Model)
```

```
Label actual:  disgust
Predicted Label: disgust
```

In [ ]:

```
test_image(700,images_f,images_f_2,Model)
```

```
Label actual:  surprise
Predicted Label: surprise
```

In [ ]:

```
test_image(900,images_f,images_f_2,Model)
```

```
Label actual:  anger
Predicted Label: anger
```

In [ ]:

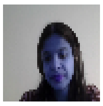
```
test_image(400,images_f,images_f_2,Model)
```

```
Label actual:  contempt
Predicted Label: contempt
```

In [ ]:

```
import pickle

filename = 'finalized_model.sav'
Model.save("/content/drive/MyDrive/emotion-model")

test_image ="/content/drive/MyDrive/test/img.png"
image=cv2.imread(test_image)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image= cv2.resize(image,(48,48))
cv2_imshow(image)
image_f = np.array(image)/255
print(image_f.shape)
pred_1=Model.predict(np.array([image_f]))
pred_class=Exp[int(np.argmax(pred_1))]
print("Predicted Label: "+ pred_class)
```

```
INFO:tensorflow:Assets written to: /content/drive/MyDrive/emotion-model/as
sets
```



```
(48, 48, 3)
Predicted Label: fear
```