In [1]:
```
import tensorflow as tf
from tensorflow.keras.layers import Dense , Flatten , Dropout , Conv2D , MaxPoolin
from tensorflow.keras.models import Model , Sequential
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from tensorflow.keras.callbacks import EarlyStopping
```

In [4]:
```
data_dir = 'dataset'
```

In [6]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(data_dir,validation
                                                               subset='training',
                                                               image_size=(224,224),
                                                               shuffle=True,
                                                               seed=123,
                                                               batch_size=32)
```

```
Found 1491 files belonging to 8 classes.
Using 1044 files for training.
```

In [7]:
```
class_labels=train_ds.class_names
class_labels
```

Out[7]:
```
['bohochic',
 'business',
 'darkacademia',
 'desi',
 'elegant',
 'grunge',
 'minimalist',
 'streetwear']
```

In [4]:
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(data_dir,validation_s
                                                               subset='validation',
                                                               image_size=(224,224),
                                                               shuffle=True,
                                                               seed=123,
                                                               batch_size=32)
```

```
Found 1491 files belonging to 8 classes.
Using 447 files for validation.
```

In [5]:
```
# example of tending the vgg16 model
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
# load model without classifier layers
# model = VGG16(include_top=False, input_shape=(224,224,3))
model=tf.keras.applications.MobileNet(
    input_shape=(224,224,3),
    alpha=1.0,
    depth_multiplier=1,
    dropout=0.001,
    include_top=False,
    weights="imagenet",
    pooling="avg"
)
# add new classifier layers
```

```
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(1024, activation='relu')(flat1)
output = Dense(8, activation='softmax')(class1)
# define new model
model = Model(inputs=model.inputs, outputs=output)
# summarize
model.summary()
# ...
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizati  (None, 112, 112, 32)      128
 on)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D  (None, 112, 112, 32)      288
 )

 conv_dw_1_bn (BatchNormali  (None, 112, 112, 32)      128
 zation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormali  (None, 112, 112, 64)      256
 zation)

 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D  (None, 56, 56, 64)        576
 )

 conv_dw_2_bn (BatchNormali  (None, 56, 56, 64)        256
 zation)

 conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

 conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

 conv_pw_2_bn (BatchNormali  (None, 56, 56, 128)       512
 zation)

 conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

 conv_dw_3 (DepthwiseConv2D  (None, 56, 56, 128)       1152
 )

 conv_dw_3_bn (BatchNormali  (None, 56, 56, 128)       512
 zation)

 conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

 conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

 conv_pw_3_bn (BatchNormali  (None, 56, 56, 128)       512
 zation)

 conv_pw_3_relu (ReLU)       (None, 56, 56, 128)       0

 conv_pad_4 (ZeroPadding2D)  (None, 57, 57, 128)       0

                             28, 28, 128)       1152
 )
```

```
conv_dw_4_bn (BatchNormali  (None, 28, 28, 128)       512
zation)

conv_dw_4_relu (ReLU)       (None, 28, 28, 128)       0

conv_pw_4 (Conv2D)          (None, 28, 28, 256)       32768

conv_pw_4_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_pw_4_relu (ReLU)       (None, 28, 28, 256)       0

conv_dw_5 (DepthwiseConv2D  (None, 28, 28, 256)       2304
)

conv_dw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_dw_5_relu (ReLU)       (None, 28, 28, 256)       0

conv_pw_5 (Conv2D)          (None, 28, 28, 256)       65536

conv_pw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_pw_5_relu (ReLU)       (None, 28, 28, 256)       0

conv_pad_6 (ZeroPadding2D)  (None, 29, 29, 256)       0

conv_dw_6 (DepthwiseConv2D  (None, 14, 14, 256)       2304
)

conv_dw_6_bn (BatchNormali  (None, 14, 14, 256)       1024
zation)

conv_dw_6_relu (ReLU)       (None, 14, 14, 256)       0

conv_pw_6 (Conv2D)          (None, 14, 14, 512)       131072

conv_pw_6_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_6_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_7 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_7_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_7 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_7_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_8 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
```

```
zation)

conv_dw_8_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_8 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_8_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_9 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_9 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_10 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_dw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_pw_10 (Conv2D)         (None, 14, 14, 512)       262144

conv_pw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_dw_11 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_11_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_dw_11_relu (ReLU)      (None, 14, 14, 512)       0

conv_pw_11 (Conv2D)         (None, 14, 14, 512)       262144

conv_pw_11_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_11_relu (ReLU)      (None, 14, 14, 512)       0

conv_pad_12 (ZeroPadding2D  (None, 15, 15, 512)       0
)

conv_dw_12 (DepthwiseConv2  (None, 7, 7, 512)         4608
D)

                            7, 7, 512)         2048
ization)
```

```
conv_dw_12_relu (ReLU)      (None, 7, 7, 512)         0

conv_pw_12 (Conv2D)         (None, 7, 7, 1024)        524288

conv_pw_12_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_pw_12_relu (ReLU)      (None, 7, 7, 1024)        0

conv_dw_13 (DepthwiseConv2  (None, 7, 7, 1024)        9216
D)

conv_dw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_dw_13_relu (ReLU)      (None, 7, 7, 1024)        0

conv_pw_13 (Conv2D)         (None, 7, 7, 1024)        1048576

conv_pw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

global_average_pooling2d (  (None, 1024)              0
GlobalAveragePooling2D)

flatten (Flatten)           (None, 1024)              0

dense (Dense)               (None, 1024)              1049600

dense_1 (Dense)             (None, 8)                 8200

=================================================================
Total params: 4286664 (16.35 MB)
Trainable params: 4264776 (16.27 MB)
Non-trainable params: 21888 (85.50 KB)
_________________________________________________________________
```

In [6]:

```
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),optimizer=opt ,
```

In [8]:

```
# Set callback functions to early stop training
# mycallbacks = [EarlyStopping(monitor='val_loss', patience=5)]
hist = model.fit(train_ds,validation_data=val_ds, epochs=20 )
```

```
Epoch 1/20
33/33 [==============================] - 94s 3s/step - loss: 0.0224 - accuracy: 0.
9943 - val_loss: 1.2115 - val_accuracy: 0.6622
Epoch 2/20
33/33 [==============================] - 91s 3s/step - loss: 0.0185 - accuracy: 0.
9943 - val_loss: 1.3176 - val_accuracy: 0.6667
Epoch 3/20
33/33 [==============================] - 88s 3s/step - loss: 0.0141 - accuracy: 0.
9933 - val_loss: 1.2977 - val_accuracy: 0.6801
Epoch 4/20
33/33 [==============================] - 94s 3s/step - loss: 0.0128 - accuracy: 0.
9943 - val_loss: 1.2712 - val_accuracy: 0.7025
Epoch 5/20
33/33 [==============================] - 84s 3s/step - loss: 0.0110 - accuracy: 0.
9952 - val_loss: 1.2860 - val_accuracy: 0.6711
Epoch 6/20
33/33 [==============================] - 84s 3s/step - loss: 0.0125 - accuracy: 0.
9923 - val_loss: 1.2893 - val_accuracy: 0.6667
Epoch 7/20
33/33 [==============================] - 84s 3s/step - loss: 0.0116 - accuracy: 0.
9923 - val_loss: 1.2691 - val_accuracy: 0.6823
Epoch 8/20
33/33 [==============================] - 84s 3s/step - loss: 0.0091 - accuracy: 0.
9943 - val_loss: 1.2950 - val_accuracy: 0.6689
Epoch 9/20
33/33 [==============================] - 85s 3s/step - loss: 0.0065 - accuracy: 0.
9981 - val_loss: 1.3099 - val_accuracy: 0.6779
Epoch 10/20
33/33 [==============================] - 86s 3s/step - loss: 0.0145 - accuracy: 0.
9943 - val_loss: 1.6950 - val_accuracy: 0.6264
Epoch 11/20
33/33 [==============================] - 90s 3s/step - loss: 0.0085 - accuracy: 0.
9962 - val_loss: 1.5311 - val_accuracy: 0.6376
Epoch 12/20
33/33 [==============================] - 87s 3s/step - loss: 0.0080 - accuracy: 0.
9943 - val_loss: 1.5053 - val_accuracy: 0.6331
Epoch 13/20
33/33 [==============================] - 84s 3s/step - loss: 0.0080 - accuracy: 0.
9952 - val_loss: 1.4708 - val_accuracy: 0.6398
Epoch 14/20
33/33 [==============================] - 83s 3s/step - loss: 0.0069 - accuracy: 0.
9962 - val_loss: 1.4100 - val_accuracy: 0.6488
Epoch 15/20
33/33 [==============================] - 89s 3s/step - loss: 0.0071 - accuracy: 0.
9943 - val_loss: 1.3687 - val_accuracy: 0.6689
Epoch 16/20
33/33 [==============================] - 88s 3s/step - loss: 0.0066 - accuracy: 0.
9962 - val_loss: 1.3727 - val_accuracy: 0.6734
Epoch 17/20
33/33 [==============================] - 86s 3s/step - loss: 0.0066 - accuracy: 0.
9962 - val_loss: 1.3516 - val_accuracy: 0.6779
Epoch 18/20
33/33 [==============================] - 85s 3s/step - loss: 0.0066 - accuracy: 0.
9952 - val_loss: 1.3523 - val_accuracy: 0.6779
Epoch 19/20
33/33 [==============================] - 84s 3s/step - loss: 0.0062 - accuracy: 0.
9952 - val_loss: 1.3639 - val_accuracy: 0.6689
Epoch 20/20
33/33 [==============================] - 86s 3s/step - loss: 0.0065 - accuracy: 0.
9943 - val_loss: 1.3766 - val_accuracy: 0.6734
```

In [9]:

```
model.save_weights("fashionstyle8mobilenetv1_67.h5")
```

In [15]:
```
from PIL import Image
# Function to preprocess an image for prediction
def preprocess_image(image_path):
    image = Image.open(image_path)
    # Preprocess the image as needed, e.g., resize, normalize, and convert to NumP
    image = image.resize((224, 224))  # Example: Resize the image to match your mo

    return image
```

In [35]:
```
test_dir='test'
```

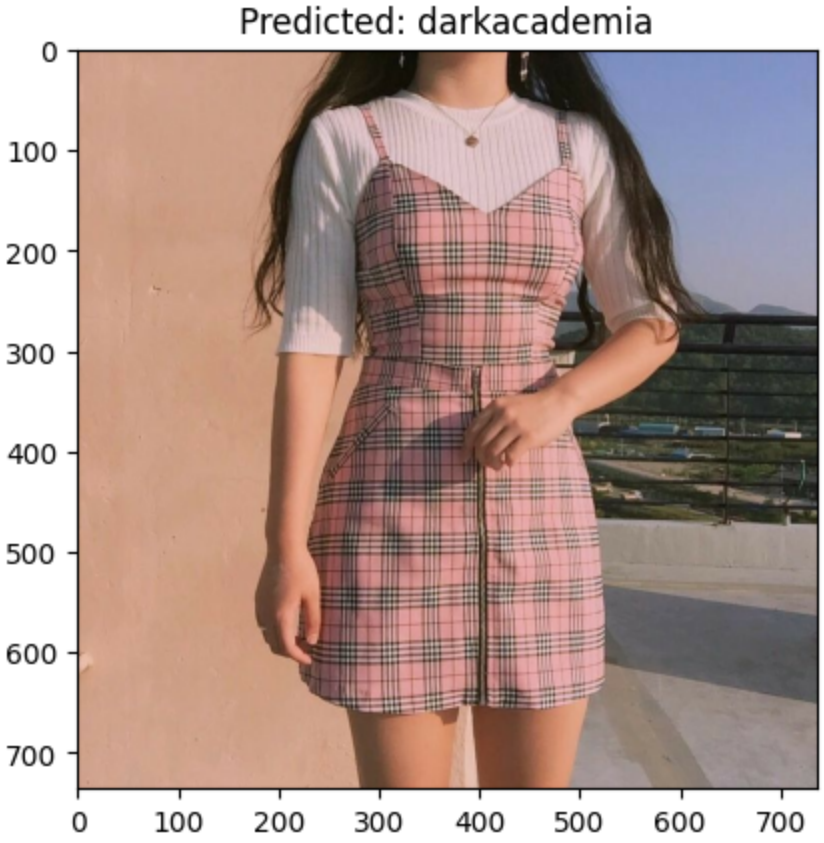
In [36]:
```
import os
import numpy as np
file_names = []
predictions = []
for filename in os.listdir(test_dir):
    if filename.endswith('.jpg') or filename.endswith('.jpeg'):
        file_path = os.path.join(test_dir, filename)
        # Preprocess the image
        processed_image = preprocess_image(file_path)
        # Expand dimensions to match the input shape expected by the model
        processed_image = np.expand_dims(processed_image, axis=0)
        # Make predictions using the model
        prediction = model.predict(processed_image)
        # Store the file name and prediction
        file_names.append(filename)
        predictions.append(prediction[0])
```

```
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 46ms/step
```

In [37]:
```
file_names,predictions
```

Out[37]:
```
(['49f7481952681610b7a4a8e1e2aa8e5b.jpg',
  '694901b813b6c1c148088328d294f754.jpg',
  '7afef5e9709418d804d215d2545c3f99.jpg',
  'Black-Flower-Design-Lehenga-Le-1651231661.jpeg',
  'download.jpeg',
  'rtnvy5niti5i70470.jpg'],
 [array([3.6339912e-10, 4.3355069e-08, 2.3050828e-09, 2.1395061e-10,
         7.5106019e-11, 5.4462244e-09, 1.4530442e-08, 1.0000000e+00],
        dtype=float32),
  array([6.1475221e-05, 5.4293359e-05, 9.5317215e-03, 5.1701363e-07,
         5.1538365e-05, 9.8039001e-01, 8.0302812e-04, 9.1073625e-03],
        dtype=float32),
  array([0.01477896, 0.08952449, 0.55665386, 0.00275655, 0.27851778,
         0.00177788, 0.05295841, 0.00303207], dtype=float32),
  array([1.2185299e-03, 1.8174995e-05, 1.6989421e-06, 9.8537236e-01,
         6.5822774e-03, 6.7464057e-03, 4.7129647e-06, 5.5748766e-05],
        dtype=float32),
  array([1.0478443e-02, 5.4405630e-04, 9.0003999e-07, 5.8360487e-01,
         4.0504006e-01, 2.1330314e-04, 7.9909565e-05, 3.8465772e-05],
        dtype=float32),
  array([9.8913017e-07, 7.3687780e-01, 3.1653656e-07, 6.6013559e-04,
         2.6235953e-01, 2.5768969e-07, 9.4186194e-05, 6.7524356e-06],
        dtype=float32)])
```

In [38]:
```
import matplotlib.pyplot as plt
for i in range(len(file_names)):
    plt.figure()
    img_path='test/'+file_names[i]
    img=plt.imread(img_path)
    plt.imshow(img.astype("uint8"))  # Display the image
    predicted_class = np.argmax(predictions[i])
    plt.title(f'Predicted: {class_labels[predicted_class]}')
    plt.show()
```



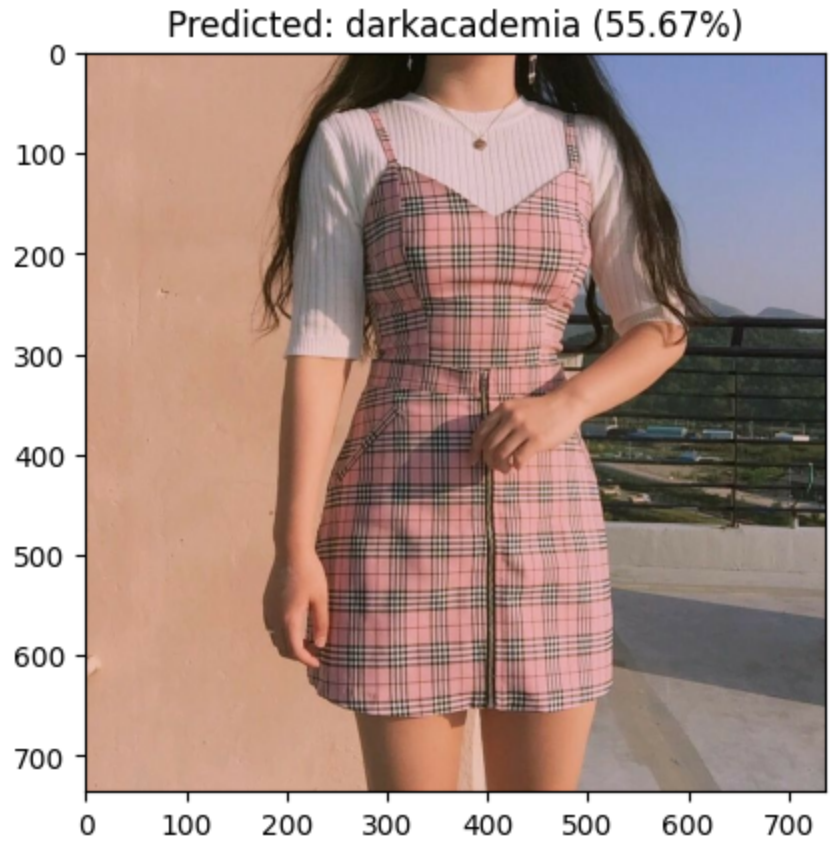
In [39]:

```
# Display the images along with the predicted class labels and percentages
for i in range(len(file_names)):
    plt.figure()
    img_path='test/'+file_names[i]
    img=plt.imread(img_path)
    plt.imshow(img.astype("uint8"))  # Display the image

    predicted_class = np.argmax(predictions[i])
    predicted_percentage = predictions[i][predicted_class] * 100
    plt.title(f'Predicted: {class_labels[predicted_class]} ({predicted_percentage:

    # Display only labels with percentage above 0
    for j in range(len(class_labels)):
        percentage = predictions[i][j] * 100
        if round(percentage) > 0.00:
            print(f'{class_labels[j]}: {percentage:.2f}%')

    plt.show()
```

```
streetwear: 100.00%
```


```
darkacademia: 0.95%
grunge: 98.04%
streetwear: 0.91%
```


```
bohochic: 1.48%
business: 8.95%
darkacademia: 55.67%
elegant: 27.85%
minimalist: 5.30%
```




```
desi: 98.54%
elegant: 0.66%
grunge: 0.67%
```


```
bohochic: 1.05%
desi: 58.36%
elegant: 40.50%
```


```
business: 73.69%
elegant: 26.24%
```

In [ ]:

In [2]:

```
# from tensorflow.keras.applications import MobileNet
# from tensorflow.keras.models import load_model
```

In [3]:
```
# base_model = MobileNet(weights='imagenet', include_top=False, input_shape=(224,
```

In [ ]:
```
# model = tf.keras.Sequential([
#     base_model,
#     tf.keras.layers.GlobalAveragePooling2D(),
#     tf.keras.layers.Dense(128, activation='relu'),
#     tf.keras.layers.Dropout(0.5),
#     tf.keras.layers.Dense(num_classes, activation='softmax')  # num_classes is t
# ])
```