In [1]:
```
import os
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models, callbacks
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Training Dataset :

.The training set contains a diverse dataset of car images of damaged vehicles from various angles, lighting conditions along with labels indicating the specific type of damage

.(e.g., dents, scratches, cracks, collision damage, etc)

images folder contains the images to be used for training the model

train.csv contains the 3 columns: image_id, filename and target class of the images present in the training dataset.

Column Description :

- image_id : Unique identifier of the image
- filename : Filename of the image
- label : Type of the damage present in the car

label column represents :

- 1: crack
- 2: scratch

- 3: tire flat
- 4: dent
- 5: glass shatter
- 6: lamp broken

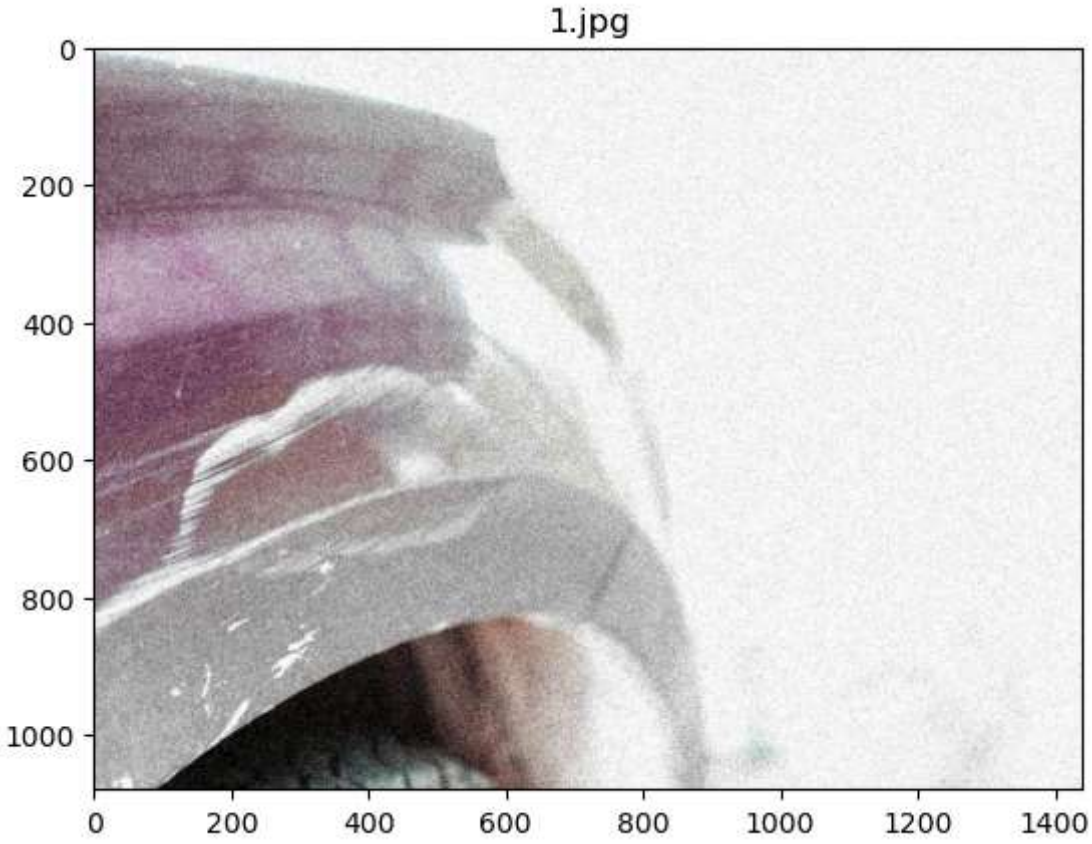
In [2]:
```
# Load training data
train_df = pd.read_csv(r"C:\Users\Ravi Kumar\Downloads\train.zip-20231201T043038Z-001\train\train\train.csv")
train_dir = "C:\\Users\\Ravi Kumar\\Downloads\\train.zip-20231201T043038Z-001\\train\\train\\images"
```

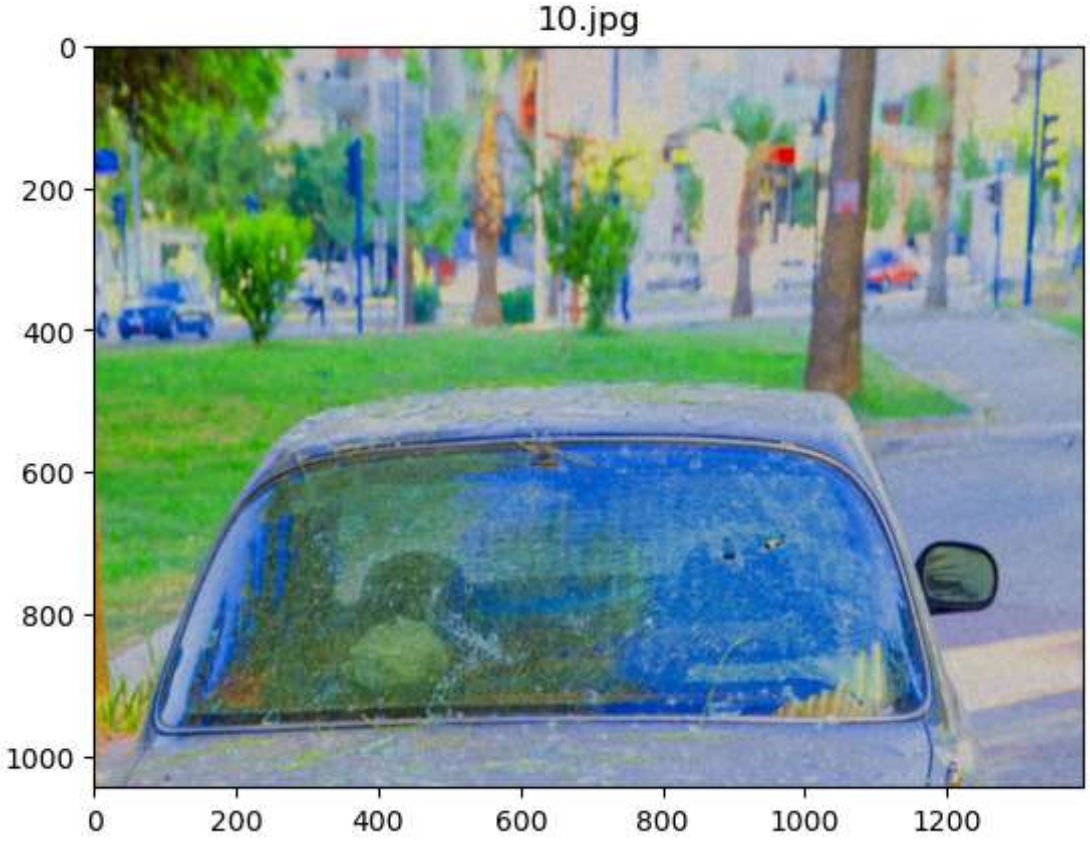
In [3]:
```
train_df.head()
```

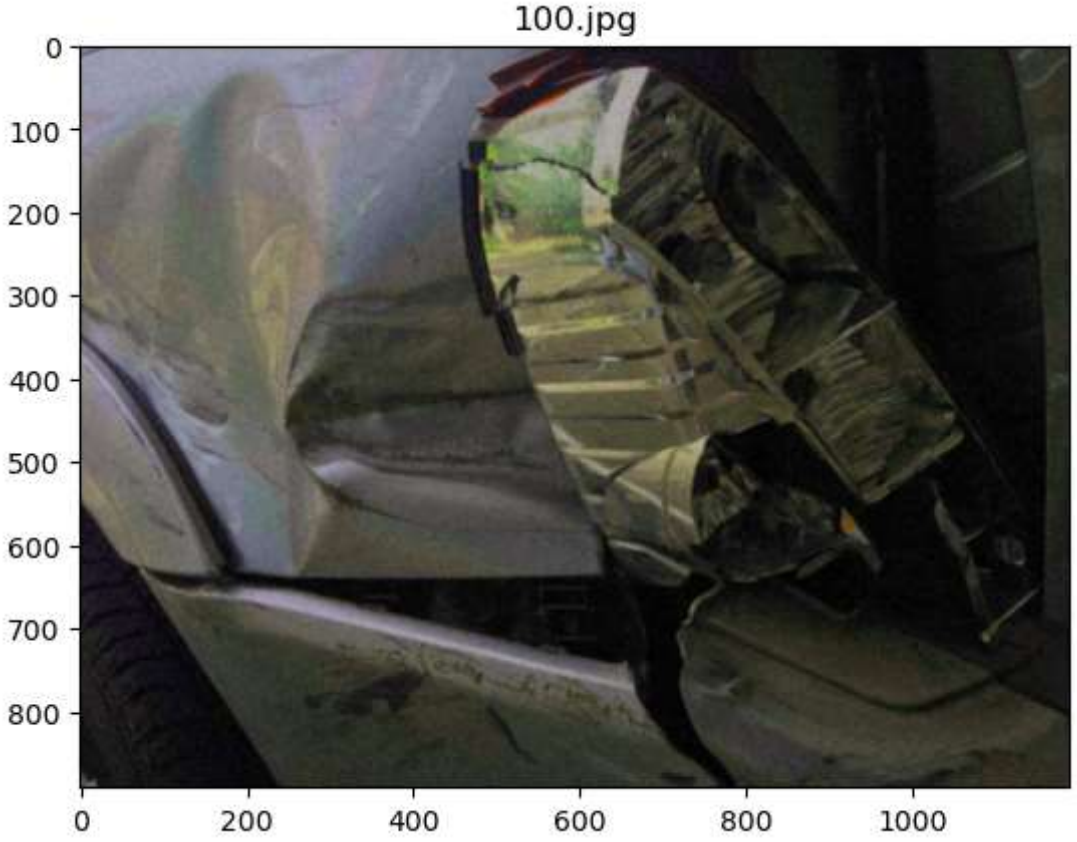
Out[3]:

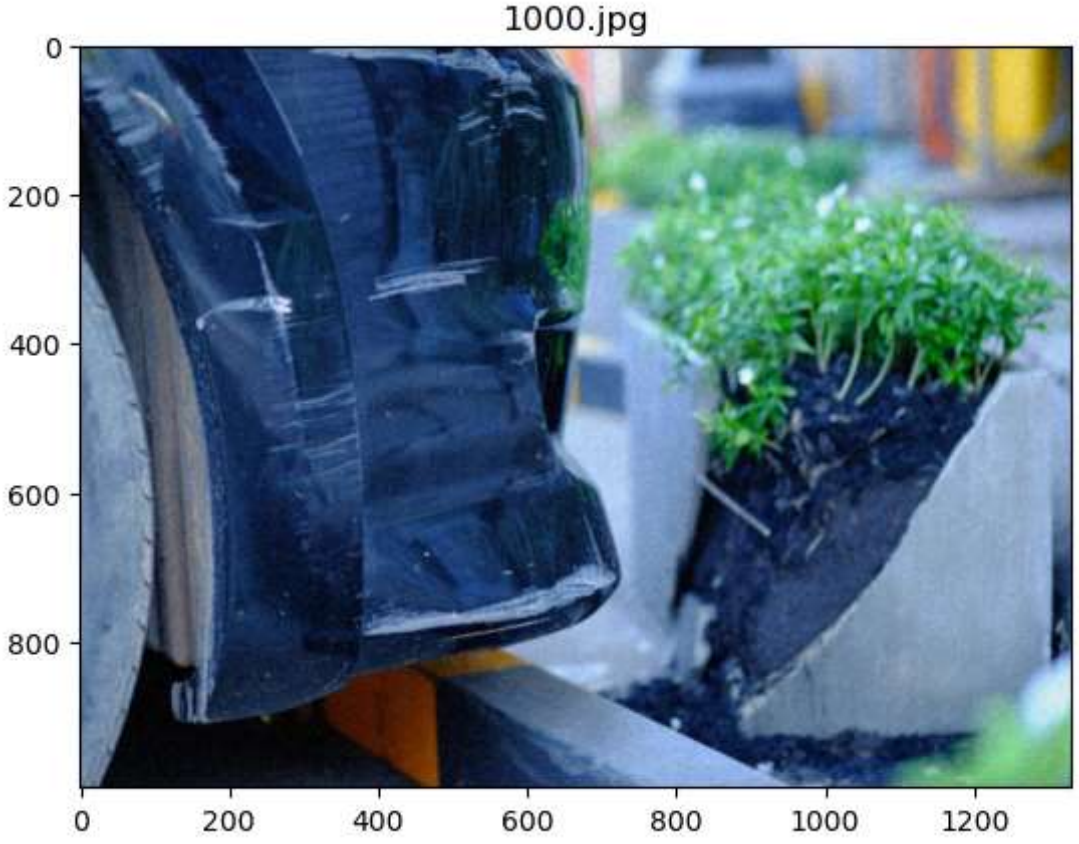
	image_id	filename	label
0	1	1.jpg	2
1	2	2.jpg	4
2	3	3.jpg	2
3	4	4.jpg	3
4	5	5.jpg	5

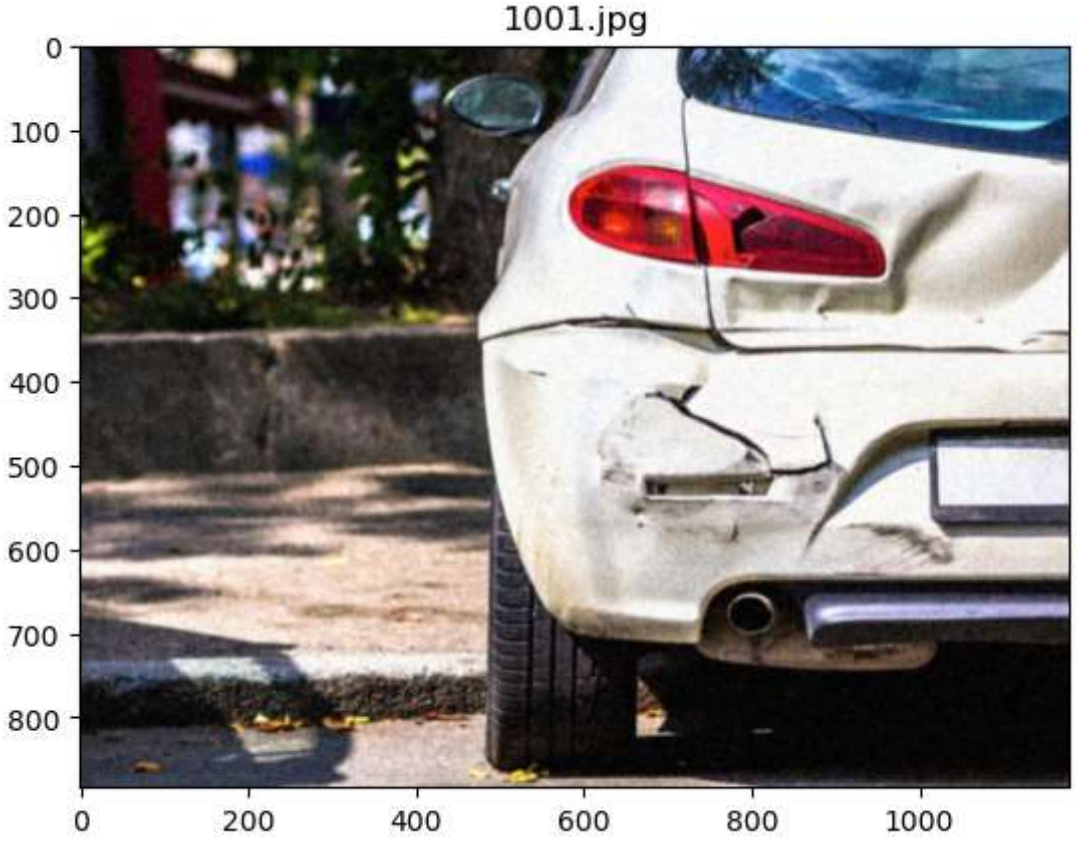
In [5]:
```
# List all files in the directory
image_files = os.listdir(train_dir)

# Display the first three images in the directory
for i, image_file in enumerate(image_files):
    # Construct the full path to the image
    image_path = os.path.join(train_dir, image_file)

    # Load and display the image
    img = mpimg.imread(image_path)
    plt.imshow(img)
    plt.title(image_file)
    plt.show()

    # Break the loop after displaying 3 images
    if i == 5:
        break
```

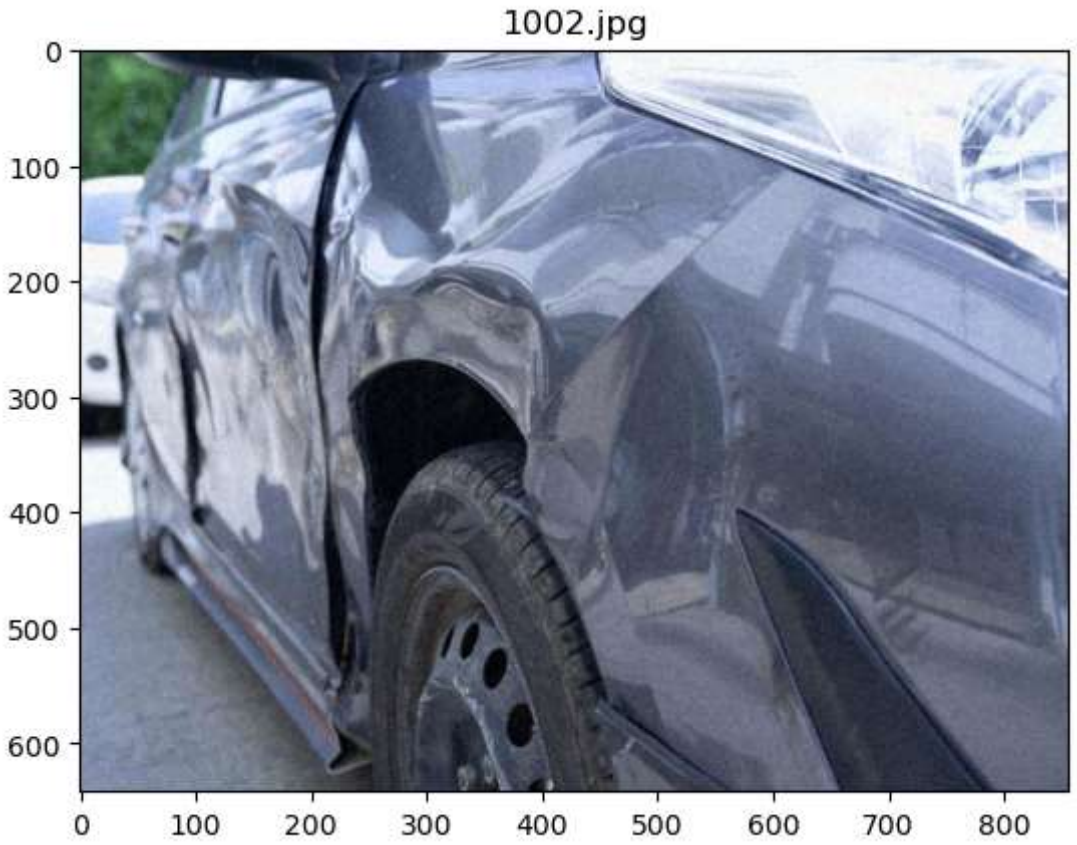

1.jpg
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


10.jpg
0
200
400
600
800
1000
0
200
400
600
800
1000
1200


100.jpg
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


1000.jpg
0
200
400
600
800
0
200
400
600
800
1000
1200


1001.jpg
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000



In [3]:
```
# Convert 'label' column to string type
train_df['label'] = train_df['label'].astype(str)
```

In [4]:
```
# Image data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

In [5]:
```
# Split the data into training and validation sets
train_data, val_data = train_test_split(train_df, test_size=0.2, random_state=42)
```

In [6]:
```
# Create data generators
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_data,
    directory=train_dir,
    x_col='filename',
    y_col='label',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 5760 validated image filenames belonging to 6 classes.
```

In [7]:
```
val_generator = train_datagen.flow_from_dataframe(
    dataframe=val_data,
    directory=train_dir,
    x_col='filename',
    y_col='label',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 1440 validated image filenames belonging to 6 classes.
```

In [8]:
```
# Build the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(6, activation='softmax')  # 6 classes for damage types
])
```

In [9]:
```
from tensorflow.keras import optimizers
# Compile the model with mini-batch SGD
sgd_optimizer = optimizers.SGD(learning_rate=0.01, momentum=0.9)  # Adjust learning rate and momentum as needed
model.compile(optimizer=sgd_optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

In [10]:
```
history = model.fit(
    train_generator,
    epochs=115,
    validation_data=val_generator
)
```

```
Epoch 1/115
180/180 [==============================] - 571s 3s/step - loss: 1.5467 - accuracy: 0.3304 - val_loss: 1.4963 - val_accu
racy: 0.3417
Epoch 2/115
180/180 [==============================] - 570s 3s/step - loss: 1.4365 - accuracy: 0.3759 - val_loss: 1.3437 - val_accu
racy: 0.4132
Epoch 3/115
180/180 [==============================] - 587s 3s/step - loss: 1.3951 - accuracy: 0.3929 - val_loss: 1.3284 - val_accu
racy: 0.4319
Epoch 4/115
180/180 [==============================] - 552s 3s/step - loss: 1.3507 - accuracy: 0.4222 - val_loss: 1.2966 - val_accu
racy: 0.4431
Epoch 5/115
180/180 [==============================] - 555s 3s/step - loss: 1.3054 - accuracy: 0.4413 - val_loss: 1.3158 - val_accu
racy: 0.4222
Epoch 6/115
180/180 [==============================] - 577s 3s/step - loss: 1.2685 - accuracy: 0.4552 - val_loss: 1.2560 - val_accu
racy: 0.4812
Epoch 7/115
180/180 [==============================] - 506s 3s/step - loss: 1.2293 - accuracy: 0.4818 - val_loss: 1.2590 - val_accu
racy: 0.4674
Epoch 8/115
180/180 [==============================] - 332s 2s/step - loss: 1.2186 - accuracy: 0.4797 - val_loss: 1.2460 - val_accu
racy: 0.4597
Epoch 9/115
180/180 [==============================] - 366s 2s/step - loss: 1.1836 - accuracy: 0.4974 - val_loss: 1.2564 - val_accu
racy: 0.4833
Epoch 10/115
180/180 [==============================] - 330s 2s/step - loss: 1.1582 - accuracy: 0.5135 - val_loss: 1.2131 - val_accu
racy: 0.4833
Epoch 11/115
180/180 [==============================] - 332s 2s/step - loss: 1.1316 - accuracy: 0.5307 - val_loss: 1.1487 - val_accu
racy: 0.5174
Epoch 12/115
180/180 [==============================] - 362s 2s/step - loss: 1.1111 - accuracy: 0.5366 - val_loss: 1.1393 - val_accu
racy: 0.5299
Epoch 13/115
180/180 [==============================] - 333s 2s/step - loss: 1.0942 - accuracy: 0.5385 - val_loss: 1.1766 - val_accu
racy: 0.5000
Epoch 14/115
180/180 [==============================] - 346s 2s/step - loss: 1.0689 - accuracy: 0.5479 - val_loss: 1.1102 - val_accu
racy: 0.5451
Epoch 15/115
180/180 [==============================] - 368s 2s/step - loss: 1.0453 - accuracy: 0.5601 - val_loss: 1.1588 - val_accu
racy: 0.5083
```

```
Epoch 16/115
180/180 [==============================] - 330s 2s/step - loss: 1.0294 - accuracy: 0.5642 - val_loss: 1.0766 - val_accu
racy: 0.5389
Epoch 17/115
180/180 [==============================] - 329s 2s/step - loss: 1.0300 - accuracy: 0.5663 - val_loss: 1.0833 - val_accu
racy: 0.5424
Epoch 18/115
180/180 [==============================] - 363s 2s/step - loss: 1.0096 - accuracy: 0.5792 - val_loss: 1.0051 - val_accu
racy: 0.5625
Epoch 19/115
180/180 [==============================] - 328s 2s/step - loss: 0.9789 - accuracy: 0.5814 - val_loss: 1.0038 - val_accu
racy: 0.5806
Epoch 20/115
180/180 [==============================] - 332s 2s/step - loss: 0.9654 - accuracy: 0.5915 - val_loss: 1.0092 - val_accu
racy: 0.5840
Epoch 21/115
180/180 [==============================] - 363s 2s/step - loss: 0.9597 - accuracy: 0.6030 - val_loss: 1.0251 - val_accu
racy: 0.5674
Epoch 22/115
180/180 [==============================] - 333s 2s/step - loss: 0.9507 - accuracy: 0.6094 - val_loss: 1.0565 - val_accu
racy: 0.5667
Epoch 23/115
180/180 [==============================] - 333s 2s/step - loss: 0.9342 - accuracy: 0.6118 - val_loss: 0.9979 - val_accu
racy: 0.6035
Epoch 24/115
180/180 [==============================] - 363s 2s/step - loss: 0.9077 - accuracy: 0.6271 - val_loss: 0.9727 - val_accu
racy: 0.5861
Epoch 25/115
180/180 [==============================] - 328s 2s/step - loss: 0.8902 - accuracy: 0.6241 - val_loss: 0.9748 - val_accu
racy: 0.6028
Epoch 26/115
180/180 [==============================] - 347s 2s/step - loss: 0.8949 - accuracy: 0.6264 - val_loss: 1.0349 - val_accu
racy: 0.5924
Epoch 27/115
180/180 [==============================] - 360s 2s/step - loss: 0.8728 - accuracy: 0.6509 - val_loss: 0.9428 - val_accu
racy: 0.6132
Epoch 28/115
180/180 [==============================] - 330s 2s/step - loss: 0.8703 - accuracy: 0.6401 - val_loss: 0.9701 - val_accu
racy: 0.5958
Epoch 29/115
180/180 [==============================] - 348s 2s/step - loss: 0.8529 - accuracy: 0.6483 - val_loss: 0.9436 - val_accu
racy: 0.5903
Epoch 30/115
180/180 [==============================] - 360s 2s/step - loss: 0.8444 - accuracy: 0.6507 - val_loss: 0.9868 - val_accu
racy: 0.6118
```

```
Epoch 31/115
180/180 [==============================] - 331s 2s/step - loss: 0.8458 - accuracy: 0.6497 - val_loss: 0.9627 - val_accu
racy: 0.6083
Epoch 32/115
180/180 [==============================] - 360s 2s/step - loss: 0.8213 - accuracy: 0.6660 - val_loss: 0.9482 - val_accu
racy: 0.5979
Epoch 33/115
180/180 [==============================] - 356s 2s/step - loss: 0.8325 - accuracy: 0.6505 - val_loss: 0.9241 - val_accu
racy: 0.6299
Epoch 34/115
180/180 [==============================] - 331s 2s/step - loss: 0.7956 - accuracy: 0.6705 - val_loss: 0.9236 - val_accu
racy: 0.6299
Epoch 35/115
180/180 [==============================] - 358s 2s/step - loss: 0.7932 - accuracy: 0.6731 - val_loss: 0.9080 - val_accu
racy: 0.6306
Epoch 36/115
180/180 [==============================] - 357s 2s/step - loss: 0.7872 - accuracy: 0.6738 - val_loss: 0.9235 - val_accu
racy: 0.6153
Epoch 37/115
180/180 [==============================] - 333s 2s/step - loss: 0.7765 - accuracy: 0.6851 - val_loss: 0.8871 - val_accu
racy: 0.6625
Epoch 38/115
180/180 [==============================] - 360s 2s/step - loss: 0.7705 - accuracy: 0.6903 - val_loss: 0.9629 - val_accu
racy: 0.6160
Epoch 39/115
180/180 [==============================] - 345s 2s/step - loss: 0.7752 - accuracy: 0.6950 - val_loss: 0.8742 - val_accu
racy: 0.6486
Epoch 40/115
180/180 [==============================] - 328s 2s/step - loss: 0.7675 - accuracy: 0.6924 - val_loss: 0.9498 - val_accu
racy: 0.6451
Epoch 41/115
180/180 [==============================] - 360s 2s/step - loss: 0.7549 - accuracy: 0.6882 - val_loss: 0.8878 - val_accu
racy: 0.6472
Epoch 42/115
180/180 [==============================] - 357s 2s/step - loss: 0.7346 - accuracy: 0.7056 - val_loss: 0.8626 - val_accu
racy: 0.6715
Epoch 43/115
180/180 [==============================] - 332s 2s/step - loss: 0.7122 - accuracy: 0.7174 - val_loss: 0.9692 - val_accu
racy: 0.6208
Epoch 44/115
180/180 [==============================] - 359s 2s/step - loss: 0.7269 - accuracy: 0.7023 - val_loss: 0.9119 - val_accu
racy: 0.6340
Epoch 45/115
180/180 [==============================] - 356s 2s/step - loss: 0.7231 - accuracy: 0.7007 - val_loss: 0.8450 - val_accu
racy: 0.6687
```

```
Epoch 46/115
180/180 [==============================] - 331s 2s/step - loss: 0.6800 - accuracy: 0.7264 - val_loss: 0.9215 - val_accu
racy: 0.6306
Epoch 47/115
180/180 [==============================] - 363s 2s/step - loss: 0.6914 - accuracy: 0.7214 - val_loss: 0.8871 - val_accu
racy: 0.6500
Epoch 48/115
180/180 [==============================] - 359s 2s/step - loss: 0.6823 - accuracy: 0.7219 - val_loss: 0.9002 - val_accu
racy: 0.6528
Epoch 49/115
180/180 [==============================] - 329s 2s/step - loss: 0.6864 - accuracy: 0.7198 - val_loss: 0.9247 - val_accu
racy: 0.6583
Epoch 50/115
180/180 [==============================] - 356s 2s/step - loss: 0.6440 - accuracy: 0.7429 - val_loss: 0.8412 - val_accu
racy: 0.6542
Epoch 51/115
180/180 [==============================] - 355s 2s/step - loss: 0.6651 - accuracy: 0.7309 - val_loss: 0.8760 - val_accu
racy: 0.6632
Epoch 52/115
180/180 [==============================] - 331s 2s/step - loss: 0.6585 - accuracy: 0.7382 - val_loss: 0.8656 - val_accu
racy: 0.6604
Epoch 53/115
180/180 [==============================] - 360s 2s/step - loss: 0.6579 - accuracy: 0.7399 - val_loss: 0.8821 - val_accu
racy: 0.6618
Epoch 54/115
180/180 [==============================] - 344s 2s/step - loss: 0.6301 - accuracy: 0.7528 - val_loss: 0.8464 - val_accu
racy: 0.6917
Epoch 55/115
180/180 [==============================] - 332s 2s/step - loss: 0.6342 - accuracy: 0.7451 - val_loss: 0.8717 - val_accu
racy: 0.6632
Epoch 56/115
180/180 [==============================] - 360s 2s/step - loss: 0.6263 - accuracy: 0.7523 - val_loss: 0.8841 - val_accu
racy: 0.6569
Epoch 57/115
180/180 [==============================] - 357s 2s/step - loss: 0.6150 - accuracy: 0.7608 - val_loss: 0.8169 - val_accu
racy: 0.6868
Epoch 58/115
180/180 [==============================] - 331s 2s/step - loss: 0.6186 - accuracy: 0.7465 - val_loss: 0.7971 - val_accu
racy: 0.6979
Epoch 59/115
180/180 [==============================] - 360s 2s/step - loss: 0.5950 - accuracy: 0.7686 - val_loss: 0.8361 - val_accu
racy: 0.6868
Epoch 60/115
180/180 [==============================] - 356s 2s/step - loss: 0.5911 - accuracy: 0.7665 - val_loss: 0.8413 - val_accu
racy: 0.6792
```

```
Epoch 61/115
180/180 [==============================] - 330s 2s/step - loss: 0.5908 - accuracy: 0.7689 - val_loss: 0.7962 - val_accu
racy: 0.6854
Epoch 62/115
180/180 [==============================] - 360s 2s/step - loss: 0.6008 - accuracy: 0.7679 - val_loss: 0.8510 - val_accu
racy: 0.6840
Epoch 63/115
180/180 [==============================] - 355s 2s/step - loss: 0.5894 - accuracy: 0.7700 - val_loss: 0.8655 - val_accu
racy: 0.6882
Epoch 64/115
180/180 [==============================] - 331s 2s/step - loss: 0.5743 - accuracy: 0.7776 - val_loss: 0.8409 - val_accu
racy: 0.6799
Epoch 65/115
180/180 [==============================] - 357s 2s/step - loss: 0.5667 - accuracy: 0.7825 - val_loss: 0.8376 - val_accu
racy: 0.6722
Epoch 66/115
180/180 [==============================] - 345s 2s/step - loss: 0.5743 - accuracy: 0.7701 - val_loss: 0.8878 - val_accu
racy: 0.6882
Epoch 67/115
180/180 [==============================] - 384s 2s/step - loss: 0.5527 - accuracy: 0.7885 - val_loss: 0.8438 - val_accu
racy: 0.6694
Epoch 68/115
180/180 [==============================] - 563s 3s/step - loss: 0.5326 - accuracy: 0.7929 - val_loss: 0.7535 - val_accu
racy: 0.7139
Epoch 69/115
180/180 [==============================] - 570s 3s/step - loss: 0.5361 - accuracy: 0.7918 - val_loss: 0.8133 - val_accu
racy: 0.6965
Epoch 70/115
180/180 [==============================] - 542s 3s/step - loss: 0.5443 - accuracy: 0.7931 - val_loss: 0.7143 - val_accu
racy: 0.7236
Epoch 71/115
180/180 [==============================] - 573s 3s/step - loss: 0.5318 - accuracy: 0.7997 - val_loss: 0.7959 - val_accu
racy: 0.7042
Epoch 72/115
180/180 [==============================] - 555s 3s/step - loss: 0.5343 - accuracy: 0.7953 - val_loss: 0.7640 - val_accu
racy: 0.7201
Epoch 73/115
180/180 [==============================] - 541s 3s/step - loss: 0.5169 - accuracy: 0.8052 - val_loss: 0.7521 - val_accu
racy: 0.7174
Epoch 74/115
180/180 [==============================] - 589s 3s/step - loss: 0.5493 - accuracy: 0.7925 - val_loss: 0.7900 - val_accu
racy: 0.7111
Epoch 75/115
180/180 [==============================] - 574s 3s/step - loss: 0.5014 - accuracy: 0.8071 - val_loss: 0.8453 - val_accu
racy: 0.7215
```

```
Epoch 76/115
180/180 [==============================] - 334s 2s/step - loss: 0.5285 - accuracy: 0.7936 - val_loss: 0.8431 - val_accu
racy: 0.6965
Epoch 77/115
180/180 [==============================] - 360s 2s/step - loss: 0.4872 - accuracy: 0.8182 - val_loss: 0.7298 - val_accu
racy: 0.7250
Epoch 78/115
180/180 [==============================] - 343s 2s/step - loss: 0.4793 - accuracy: 0.8153 - val_loss: 0.7473 - val_accu
racy: 0.7208
Epoch 79/115
180/180 [==============================] - 332s 2s/step - loss: 0.5221 - accuracy: 0.8014 - val_loss: 0.7232 - val_accu
racy: 0.7347
Epoch 80/115
180/180 [==============================] - 358s 2s/step - loss: 0.5030 - accuracy: 0.8023 - val_loss: 0.7664 - val_accu
racy: 0.7319
Epoch 81/115
180/180 [==============================] - 353s 2s/step - loss: 0.5105 - accuracy: 0.8028 - val_loss: 0.7398 - val_accu
racy: 0.7382
Epoch 82/115
180/180 [==============================] - 329s 2s/step - loss: 0.4856 - accuracy: 0.8146 - val_loss: 0.7417 - val_accu
racy: 0.7347
Epoch 83/115
180/180 [==============================] - 361s 2s/step - loss: 0.4751 - accuracy: 0.8227 - val_loss: 0.7440 - val_accu
racy: 0.7250
Epoch 84/115
180/180 [==============================] - 356s 2s/step - loss: 0.4644 - accuracy: 0.8243 - val_loss: 0.6906 - val_accu
racy: 0.7424
Epoch 85/115
180/180 [==============================] - 330s 2s/step - loss: 0.4672 - accuracy: 0.8262 - val_loss: 0.8054 - val_accu
racy: 0.7264
Epoch 86/115
180/180 [==============================] - 359s 2s/step - loss: 0.4455 - accuracy: 0.8302 - val_loss: 0.7512 - val_accu
racy: 0.7292
Epoch 87/115
180/180 [==============================] - 357s 2s/step - loss: 0.4538 - accuracy: 0.8332 - val_loss: 0.7703 - val_accu
racy: 0.7361
Epoch 88/115
180/180 [==============================] - 333s 2s/step - loss: 0.4768 - accuracy: 0.8161 - val_loss: 0.7344 - val_accu
racy: 0.7431
Epoch 89/115
180/180 [==============================] - 361s 2s/step - loss: 0.4706 - accuracy: 0.8236 - val_loss: 0.8482 - val_accu
racy: 0.7090
Epoch 90/115
180/180 [==============================] - 355s 2s/step - loss: 0.4780 - accuracy: 0.8217 - val_loss: 0.8053 - val_accu
racy: 0.7417
```

```
Epoch 91/115
180/180 [==============================] - 329s 2s/step - loss: 0.4843 - accuracy: 0.8241 - val_loss: 0.7124 - val_accu
racy: 0.7396
Epoch 92/115
180/180 [==============================] - 356s 2s/step - loss: 0.4479 - accuracy: 0.8328 - val_loss: 0.7792 - val_accu
racy: 0.7181
Epoch 93/115
180/180 [==============================] - 359s 2s/step - loss: 0.4528 - accuracy: 0.8344 - val_loss: 0.7035 - val_accu
racy: 0.7611
Epoch 94/115
180/180 [==============================] - 331s 2s/step - loss: 0.4512 - accuracy: 0.8280 - val_loss: 0.8041 - val_accu
racy: 0.7285
Epoch 95/115
180/180 [==============================] - 362s 2s/step - loss: 0.4319 - accuracy: 0.8417 - val_loss: 0.7498 - val_accu
racy: 0.7410
Epoch 96/115
180/180 [==============================] - 360s 2s/step - loss: 0.4303 - accuracy: 0.8328 - val_loss: 0.7690 - val_accu
racy: 0.7306
Epoch 97/115
180/180 [==============================] - 330s 2s/step - loss: 0.4125 - accuracy: 0.8443 - val_loss: 0.7340 - val_accu
racy: 0.7514
Epoch 98/115
180/180 [==============================] - 361s 2s/step - loss: 0.4174 - accuracy: 0.8448 - val_loss: 0.8368 - val_accu
racy: 0.7236
Epoch 99/115
180/180 [==============================] - 358s 2s/step - loss: 0.4728 - accuracy: 0.8273 - val_loss: 0.6954 - val_accu
racy: 0.7444
Epoch 100/115
180/180 [==============================] - 328s 2s/step - loss: 0.4388 - accuracy: 0.8424 - val_loss: 0.6097 - val_accu
racy: 0.7736
Epoch 101/115
180/180 [==============================] - 356s 2s/step - loss: 0.4106 - accuracy: 0.8488 - val_loss: 0.6832 - val_accu
racy: 0.7542
Epoch 102/115
180/180 [==============================] - 354s 2s/step - loss: 0.3828 - accuracy: 0.8554 - val_loss: 0.7099 - val_accu
racy: 0.7521
Epoch 103/115
180/180 [==============================] - 330s 2s/step - loss: 0.4351 - accuracy: 0.8425 - val_loss: 0.7120 - val_accu
racy: 0.7576
Epoch 104/115
180/180 [==============================] - 356s 2s/step - loss: 0.4365 - accuracy: 0.8410 - val_loss: 0.6658 - val_accu
racy: 0.7514
Epoch 105/115
180/180 [==============================] - 356s 2s/step - loss: 0.4174 - accuracy: 0.8431 - val_loss: 0.7430 - val_accu
racy: 0.7424
```

```
Epoch 106/115
180/180 [==============================] - 330s 2s/step - loss: 0.3852 - accuracy: 0.8571 - val_loss: 0.7017 - val_accu
racy: 0.7764
Epoch 107/115
180/180 [==============================] - 360s 2s/step - loss: 0.4066 - accuracy: 0.8483 - val_loss: 0.6749 - val_accu
racy: 0.7563
Epoch 108/115
180/180 [==============================] - 343s 2s/step - loss: 0.3872 - accuracy: 0.8549 - val_loss: 0.6637 - val_accu
racy: 0.7799
Epoch 109/115
180/180 [==============================] - 330s 2s/step - loss: 0.4239 - accuracy: 0.8458 - val_loss: 0.7057 - val_accu
racy: 0.7722
Epoch 110/115
180/180 [==============================] - 357s 2s/step - loss: 0.3893 - accuracy: 0.8578 - val_loss: 0.7199 - val_accu
racy: 0.7576
Epoch 111/115
180/180 [==============================] - 358s 2s/step - loss: 0.3766 - accuracy: 0.8557 - val_loss: 0.7603 - val_accu
racy: 0.7500
Epoch 112/115
180/180 [==============================] - 331s 2s/step - loss: 0.4052 - accuracy: 0.8495 - val_loss: 0.6752 - val_accu
racy: 0.7569
Epoch 113/115
180/180 [==============================] - 357s 2s/step - loss: 0.4175 - accuracy: 0.8484 - val_loss: 0.7112 - val_accu
racy: 0.7604
Epoch 114/115
180/180 [==============================] - 356s 2s/step - loss: 0.3959 - accuracy: 0.8540 - val_loss: 0.6720 - val_accu
racy: 0.7681
Epoch 115/115
180/180 [==============================] - 332s 2s/step - loss: 0.3511 - accuracy: 0.8694 - val_loss: 0.6582 - val_accu
racy: 0.7576
```

In [11]:

```python
# Evaluate the model on the validation set
val_predictions = model.predict(val_generator)
val_predicted_labels = val_predictions.argmax(axis=1) + 1  # Add 1 to match the class labels (1-indexed)

val_true_labels = val_data['label'].astype(int).values  # Convert true labels to int for comparison

# Calculate macro F1 score
macro_f1 = f1_score(val_true_labels, val_predicted_labels, average='macro')
print(f'Macro F1 Score on Validation Set: {macro_f1}')
```

```
45/45 [==============================] - 89s 2s/step
Macro F1 Score on Validation Set: 0.17838150345713288
```

Test Dataset :

In the test set, you are provided with only the images and you need to predict the type of damage for each image present in the test set.

Dataset Description :

images folder contains all the test images for which the prediction is to be done.

test.csv contains 2 columns: image_id and filename and we need to predict the label for each present in the test set.

Column Description :

- image_id : Unique identifier of the image
- filename : Filename of the image

- 'we need to predict the label for each present in the test set'

In [6]:
```
# Make predictions on the test set
test_df = pd.read_csv(r"C:\Users\Ravi Kumar\Downloads\test\test\test.csv")
test_dir = "C:\\Users\\Ravi Kumar\\Downloads\\test\\test\\images"
```

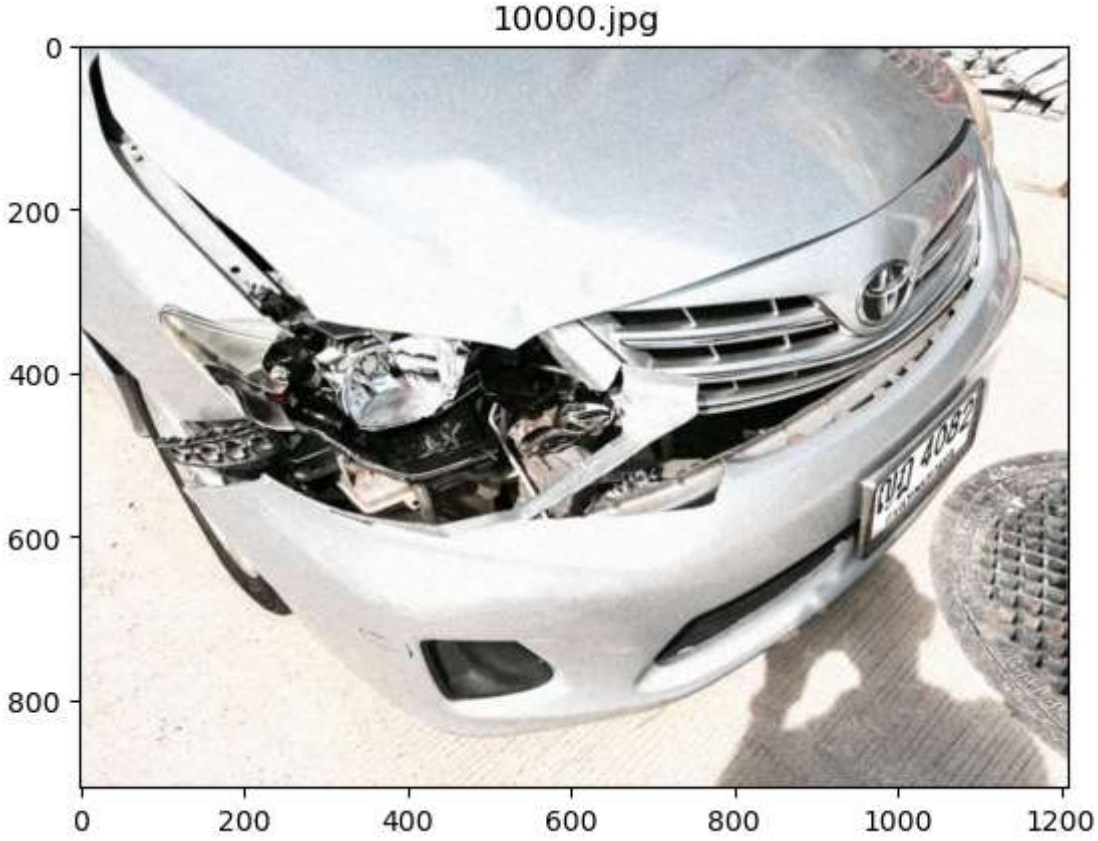
In [7]:
```
test_df.head()
```

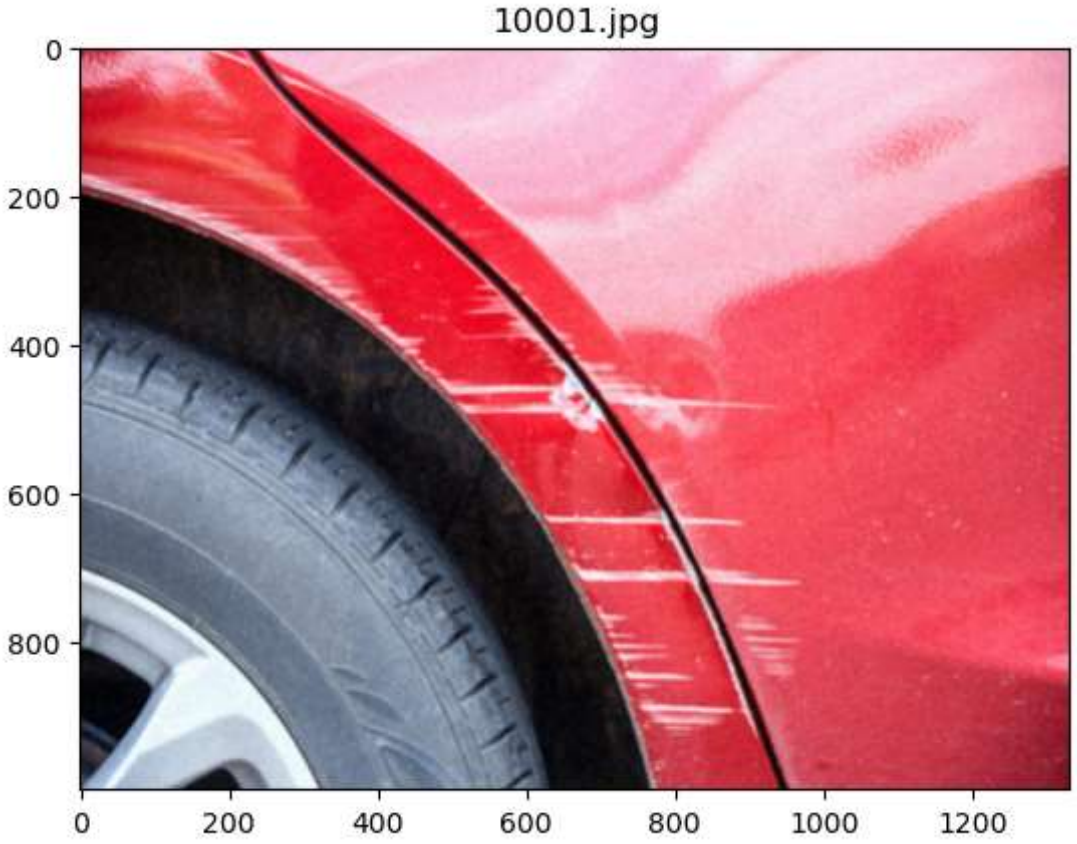
Out[7]:

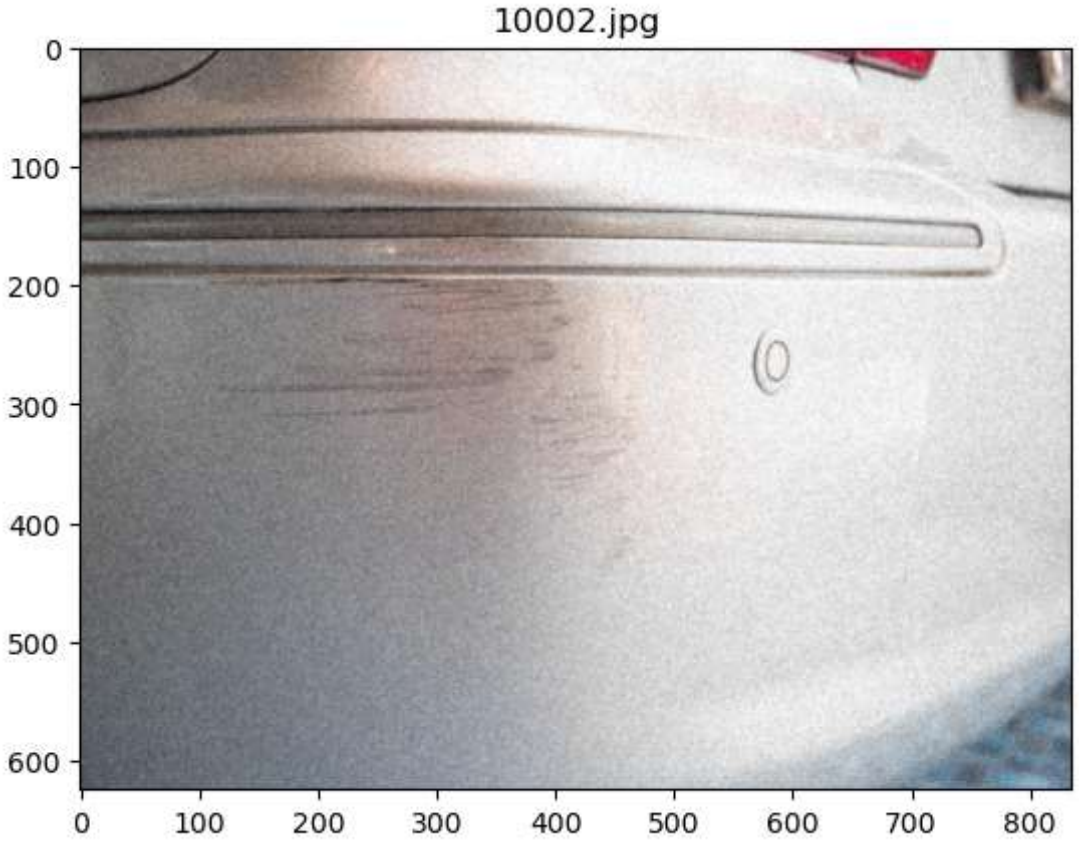
	image_id	filename
0	7201	7201.jpg
1	7202	7202.jpg
2	7203	7203.jpg
3	7204	7204.jpg
4	7205	7205.jpg

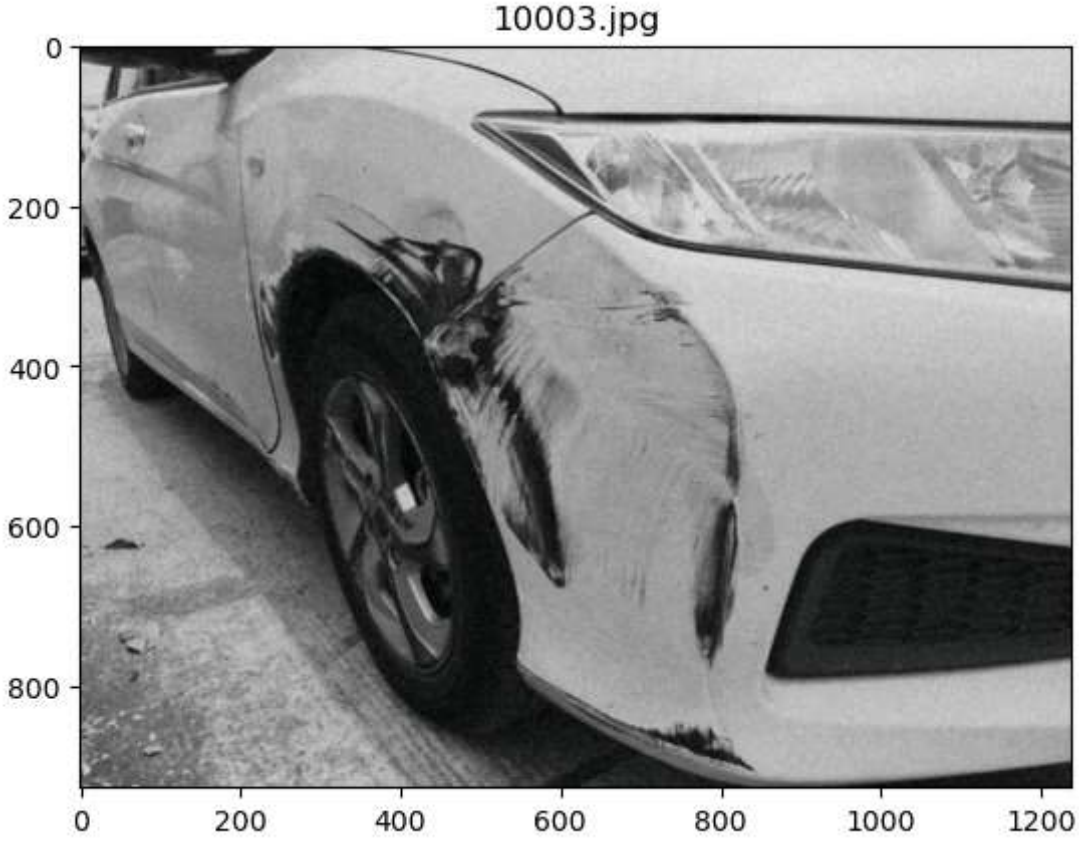
In [8]:

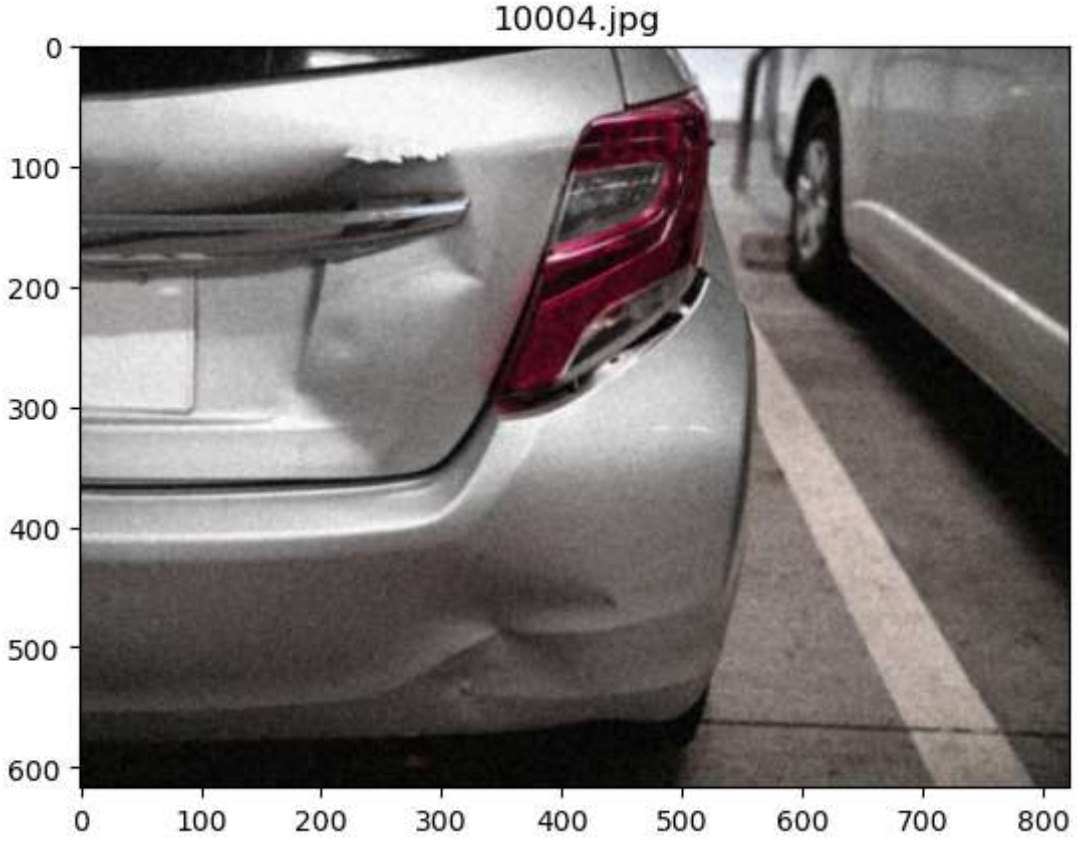
```
# List all files in the directory
test_image_files = os.listdir(test_dir)

# Display the first two images in the test directory
for i, test_image_file in enumerate(test_image_files):
    # Construct the full path to the test image
    test_image_path = os.path.join(test_dir, test_image_file)

    # Load and display the test image
    test_img = mpimg.imread(test_image_path)
    plt.imshow(test_img)
    plt.title(test_image_file)
    plt.show()

    # Break the loop after displaying 2 images
    if i == 5:
        break
```

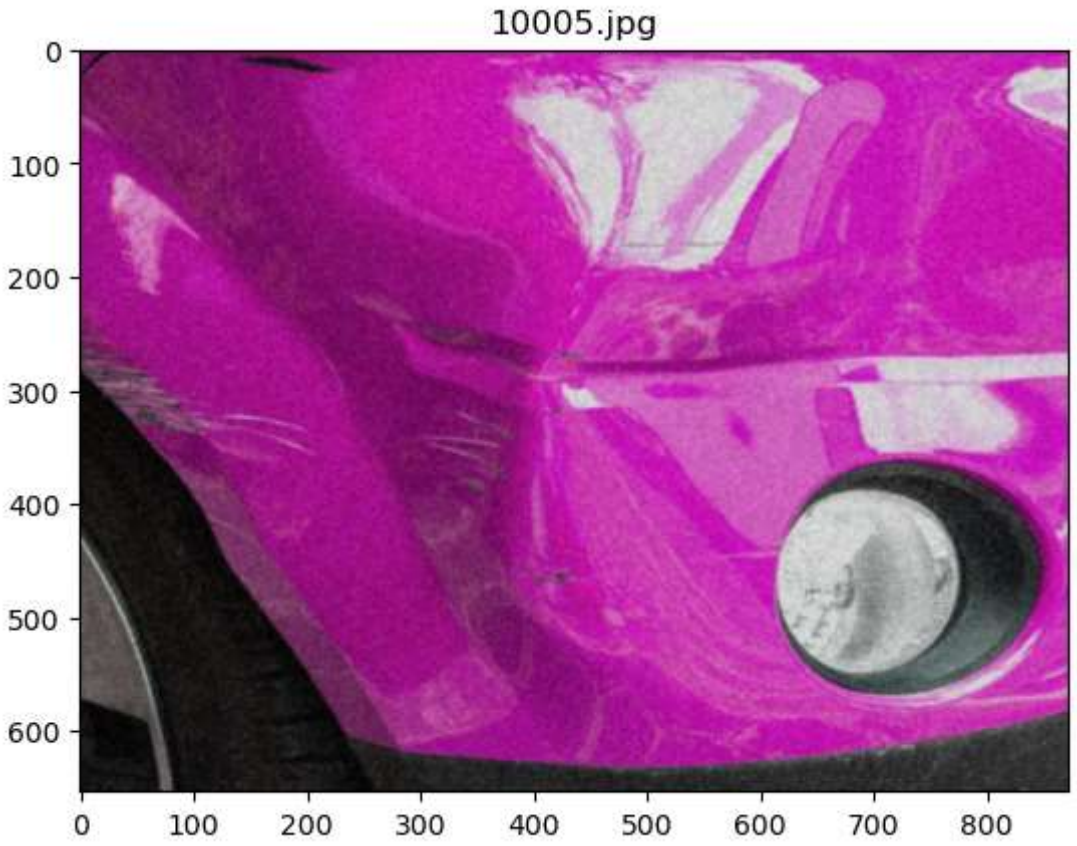

10000.jpg
0
200
400
600
800
0
200
400
600
800
1000
1200


10001.jpg
0
200
400
600
800
0
200
400
600
800
1000
1200


10002.jpg
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800


10003.jpg
0
200
400
600
800
0
200
400
600
800
1000
1200


10004.jpg
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800



In [13]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [14]:
```
test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=test_dir,
    x_col='filename',
    y_col=None,
    target_size=(224, 224),
    batch_size=32,
    class_mode=None,
    shuffle=False
)
```

```
Found 4800 validated image filenames.
```

In [15]:
```
predictions = model.predict(test_generator)
predicted_labels = predictions.argmax(axis=1) + 1  # Add 1 to match the class labels (1-indexed)
```

```
150/150 [==============================] - 193s 1s/step
```

In [23]:
```python
# Create submission file
submission_df = pd.DataFrame({'image_id': test_df['image_id'], 'label': predicted_labels})
submission_df.to_csv("C:\\Users\\Ravi Kumar\\OneDrive\\Documents\\Ripik.AI Hackfest\\predicted_data.csv", index=False)
```

In [9]:
```python
# loading predicted data set to calculate the F1 score
true_labels_df = pd.read_csv(r"C:\Users\Ravi Kumar\OneDrive\Documents\Ripik.AI Hackfest\predicted_data.csv")
```

In [33]:
```python
# Extract true labels and predicted labels
true_labels = true_labels_df['label'].astype(int).values
predicted_labels = predictions.argmax(axis=1) + 1
```

In [34]:
```python
# Calculate macro F1 score
macro_f1_test = f1_score(true_labels, predicted_labels, average='macro')
print(f'Macro F1 Score on Test Set: {macro_f1_test}')
```

```
Macro F1 Score on Test Set: 1.0
```

In [ ]:
```python

```