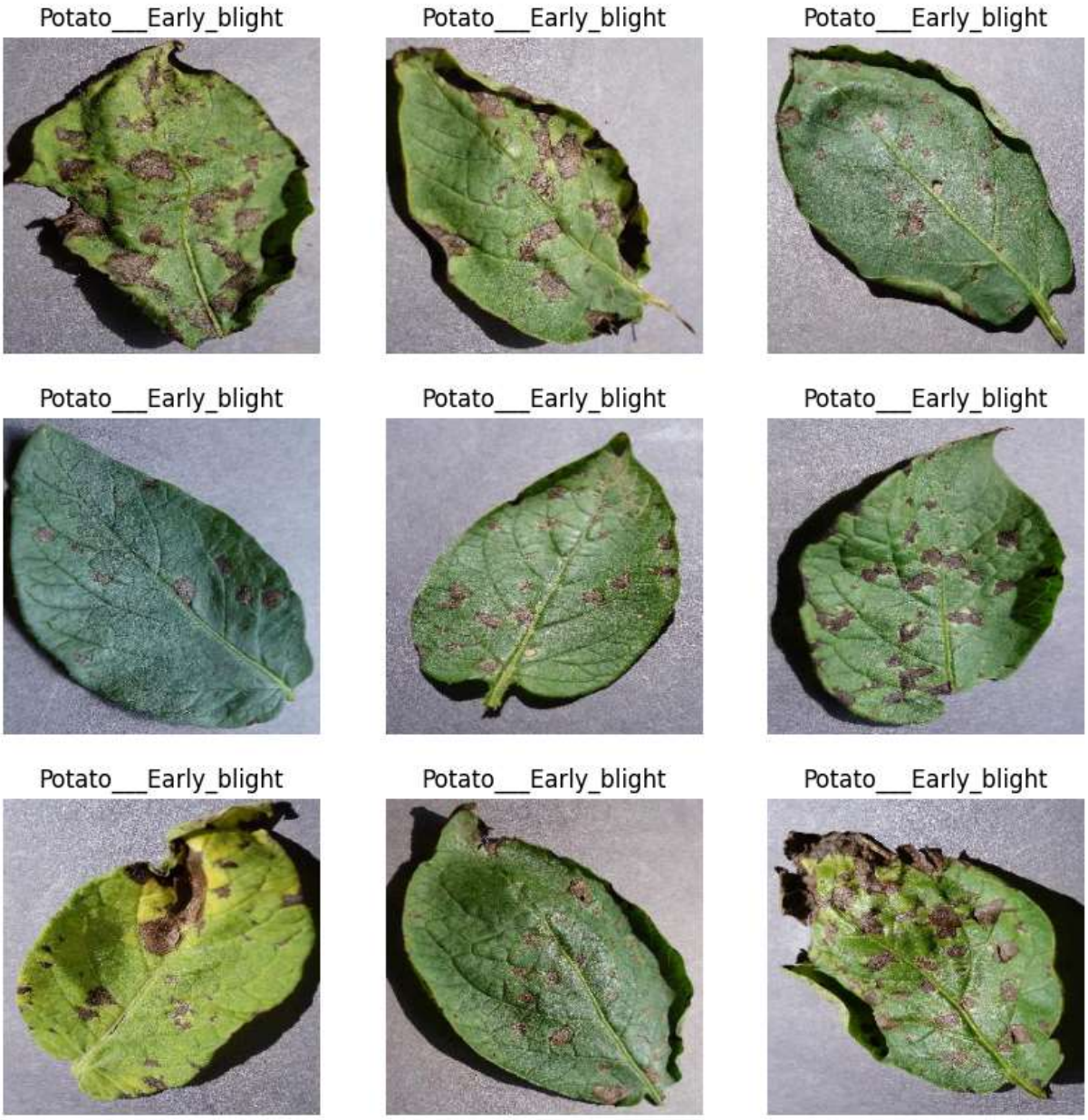
In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

In [8]:
```
BATCH_SIZE = 32
IMAGE_SIZE = 180
CHANNELS = 3
train_val_dir = r"C:\Users\MANICKA MEENAKSHI.S\Downloads\archive (6)\PotatoPlants"
test_dir = r"C:\Users\MANICKA MEENAKSHI.S\Downloads\archive (6)\PlantVillage\PlantV
```

In [17]:
```
import os
data = []

for class_name in os.listdir(train_val_dir):
    class_path = os.path.join(train_val_dir, class_name)


    if os.path.isdir(class_path):
        for img_name in os.listdir(class_path):
            img_path = os.path.join(class_path, img_name)
            data.append((img_path, class_name))


df = pd.DataFrame(data, columns=['image_path', 'label'])


print(df.head())
```

```
                                          image_path                 label
0  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
1  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
2  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
3  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
4  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
```

In [21]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg


class_names = df['label'].unique()

plt.figure(figsize=(10, 10))


for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    img_path = df['image_path'].iloc[i]
    img = mpimg.imread(img_path)
    plt.imshow(img)
    plt.title(df['label'].iloc[i])
    plt.axis("off")

plt.show()
```



In [66]:
```
import seaborn as sns
from sklearn.model_selection import train_test_split

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout,
```

In [25]:
```
import os
data = []
for class_name in os.listdir(test_dir):
    class_path = os.path.join(test_dir, class_name)
    if os.path.isdir(class_path):
        for img_name in os.listdir(class_path):
            img_path = os.path.join(class_path, img_name)
            data.append((img_path, class_name))
```

```
test = pd.DataFrame(data, columns=['image_path', 'label'])
print(test.head())
```

```
                                          image_path                 label
0  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
1  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
2  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
3  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
4  C:\Users\MANICKA MEENAKSHI.S\Downloads\archive...  Potato___Early_blight
```

In [32]:
```
from sklearn.model_selection import train_test_split

def get_dataset_partitions(df, train_split=0.8, val_split=0.1, test_split=0.1):
    assert (train_split + val_split + test_split) == 1
    train_df, temp_df = train_test_split(df, test_size=(1 - train_split), random_st
    val_size = val_split / (val_split + test_split)
    val_df, test_df = train_test_split(temp_df, test_size=(1 - val_size), random_st

    return train_df, val_df, test_df


train_df, val_df, test_df = get_dataset_partitions(df)


print(f"Train set size: {len(train_df)}")
print(f"Validation set size: {len(val_df)}")
print(f"Test set size: {len(test_df)}")
```

```
Train set size: 1721
Validation set size: 215
Test set size: 216
```

In [36]:
```
from tensorflow.keras import layers, Sequential
num_classes = len(class_names)

model = Sequential([
  layers.Rescaling(1./255, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

In [37]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [38]:
```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	
rescaling_2 (Rescaling)	(None, 180, 180, 3)	
conv2d_6 (Conv2D)	(None, 180, 180, 16)	
max_pooling2d_6 (MaxPooling2D)	(None, 90, 90, 16)	
conv2d_7 (Conv2D)	(None, 90, 90, 32)	
max_pooling2d_7 (MaxPooling2D)	(None, 45, 45, 32)	
conv2d_8 (Conv2D)	(None, 45, 45, 64)	
max_pooling2d_8 (MaxPooling2D)	(None, 22, 22, 64)	
flatten_2 (Flatten)	(None, 30976)	
dense_4 (Dense)	(None, 128)	
dense_5 (Dense)	(None, 3)	

Total params: 3,989,027 (15.22 MB)

Trainable params: 3,989,027 (15.22 MB)

Non-trainable params: 0 (0.00 B)

In [46]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rescale=1./255)
train_generator = datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

```
Found 1721 validated image filenames belonging to 3 classes.
```

In [43]:
```
print(train_df.columns)
```

```
Index(['image_path', 'label'], dtype='object')
```

In [48]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam


BATCH_SIZE = 32
```

```
datagen_train = ImageDataGenerator(rescale=1./255)
datagen_val = ImageDataGenerator(rescale=1./255)


train_generator = datagen_train.flow_from_dataframe(
    dataframe=train_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='sparse'
)

validation_generator = datagen_val.flow_from_dataframe(
    dataframe=val_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='sparse'
)

# Define the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(180, 180, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=10,
    verbose=1
)
```

```
Found 1721 validated image filenames belonging to 3 classes.
Found 215 validated image filenames belonging to 3 classes.
```

```
C:\Users\MANICKA MEENAKSHI.S\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
C:\Users\MANICKA MEENAKSHI.S\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
54/54 ━━━━━━━━━━━━━━━━━━━━ 70s 1s/step - accuracy: 0.6002 - loss: 0.9188 - val_accuracy: 0.8651 - val_loss: 0.3309
Epoch 2/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 62s 1s/step - accuracy: 0.8935 - loss: 0.2885 - val_accuracy: 0.7535 - val_loss: 0.4649
Epoch 3/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 62s 1s/step - accuracy: 0.9058 - loss: 0.2544 - val_accuracy: 0.8512 - val_loss: 0.3337
Epoch 4/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 64s 1s/step - accuracy: 0.9004 - loss: 0.2805 - val_accuracy: 0.9581 - val_loss: 0.1029
Epoch 5/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 61s 1s/step - accuracy: 0.9661 - loss: 0.0937 - val_accuracy: 0.9442 - val_loss: 0.1457
Epoch 6/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 61s 1s/step - accuracy: 0.9786 - loss: 0.0626 - val_accuracy: 0.9302 - val_loss: 0.1690
Epoch 7/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 62s 1s/step - accuracy: 0.9814 - loss: 0.0537 - val_accuracy: 0.9581 - val_loss: 0.0988
Epoch 8/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 65s 1s/step - accuracy: 0.9903 - loss: 0.0420 - val_accuracy: 0.9674 - val_loss: 0.0807
Epoch 9/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 111s 2s/step - accuracy: 0.9923 - loss: 0.0185 - val_accuracy: 0.9395 - val_loss: 0.1695
Epoch 10/10
54/54 ━━━━━━━━━━━━━━━━━━━━ 29s 528ms/step - accuracy: 0.9888 - loss: 0.0293 - val_accuracy: 0.9395 - val_loss: 0.1583
```

In [50]:

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen_test = ImageDataGenerator(rescale=1./255)


test_generator = datagen_test.flow_from_dataframe(
    dataframe=test_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='sparse'
```

```
)
score = model.evaluate(test_generator, verbose=1)

print(f"Test Loss: {score[0]}")
print(f"Test Accuracy: {score[1]}")
```

```
Found 216 validated image filenames belonging to 3 classes.
C:\Users\MANICKA MEENAKSHI.S\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 319ms/step - accuracy: 0.9645 - loss: 0.1339
Test Loss: 0.17255938053131104
Test Accuracy: 0.9583333134651184
```

In [51]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [53]:
```
datagen_val = ImageDataGenerator(rescale=1./255)

val_generator = datagen_val.flow_from_dataframe(
    dataframe=val_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
predictions = model.predict(val_generator, verbose=1)
predicted_classes = np.argmax(predictions, axis=1)

print(predictions)
print(predicted_classes)
```

```
Found 215 validated image filenames belonging to 3 classes.
C:\Users\MANICKA MEENAKSHI.S\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 382ms/step
[[2.89817952e-04 9.94333982e-01 5.37624117e-03]
 [1.00000000e+00 1.25434333e-11 7.82352386e-18]
 [8.37998826e-08 9.99999881e-01 3.39474959e-09]
 [9.99262750e-01 7.37285649e-04 1.90554495e-12]
 [6.63156925e-08 9.99999881e-01 7.49225215e-09]
 [9.99489069e-01 5.05559612e-04 5.39534358e-06]
 [2.76853348e-06 9.99994874e-01 2.30076967e-06]
 [3.03665888e-06 9.99996424e-01 5.54135340e-07]
 [6.07621314e-06 9.99993801e-01 1.01974045e-07]
 [3.52587178e-03 9.96465087e-01 8.99530278e-06]
 [8.53969553e-08 9.99999762e-01 1.80824856e-07]
 [1.72557593e-05 7.16304541e-01 2.83678234e-01]
 [2.84075355e-07 9.99999762e-01 1.12099370e-08]
 [1.96901055e-07 2.25164220e-02 9.77483392e-01]
 [2.56098778e-04 9.99742210e-01 1.70564215e-06]
 [1.91582076e-04 9.34476972e-01 6.53314218e-02]
 [9.99994040e-01 5.93455934e-06 7.05669023e-10]
 [1.00000000e+00 9.85497639e-09 5.26800537e-15]
 [1.00000000e+00 1.30392959e-08 4.06682710e-14]
 [3.60619447e-06 9.98607099e-01 1.38931058e-03]
 [9.99999881e-01 8.81933460e-08 1.28517608e-09]
 [2.05485122e-07 1.72616914e-01 8.27382922e-01]
 [1.00000000e+00 7.07014037e-13 1.10094185e-20]
 [9.97782767e-01 2.21728929e-03 9.75456087e-12]
 [3.03339812e-06 9.99994874e-01 2.11219094e-06]
 [1.00000000e+00 1.25723676e-11 7.75443826e-19]
 [1.92269963e-05 9.99971867e-01 8.93649030e-06]
 [1.09029577e-06 9.99998689e-01 2.15558984e-07]
 [3.69088480e-06 9.99996305e-01 1.28628963e-09]
 [9.99999881e-01 1.46513131e-07 1.94936973e-17]
 [3.24754723e-07 9.99999285e-01 4.15413098e-07]
 [1.54852930e-08 8.50359425e-02 9.14964020e-01]
 [4.07161082e-07 9.99936223e-01 6.33507298e-05]
 [1.37110305e-07 9.99999642e-01 2.12335081e-07]
 [1.42617000e-05 9.99762833e-01 2.22939518e-04]
 [8.13996303e-06 9.99991775e-01 1.12225166e-07]
 [2.18864501e-04 9.99781191e-01 2.42650877e-09]
 [9.93842840e-01 6.15692837e-03 2.73417271e-07]
 [1.27739241e-07 9.99999046e-01 8.04733986e-07]
 [1.83499092e-03 9.98144150e-01 2.08767342e-05]
 [9.99502659e-01 4.97317524e-04 4.34230074e-09]
 [1.00000000e+00 3.17324964e-12 8.29976386e-21]
 [1.00000000e+00 1.65305902e-09 5.95009750e-18]
 [1.00000000e+00 4.97277119e-15 3.49570344e-28]
 [4.16858727e-03 9.95831430e-01 5.60176683e-10]
 [9.99810159e-01 1.89479455e-04 3.52667428e-07]
 [1.00000000e+00 5.02427113e-08 1.66262827e-16]
 [5.19407763e-07 9.99859452e-01 1.39971657e-04]
 [4.63277452e-08 7.88508728e-02 9.21149075e-01]
 [9.82575898e-07 9.99992013e-01 6.98919848e-06]
 [4.52138345e-08 9.99998569e-01 1.44002661e-06]
 [9.99948263e-01 5.17087865e-05 1.65050279e-10]
 [1.00000000e+00 1.21500088e-10 1.46479803e-16]
 [9.39404856e-08 9.99996781e-01 3.13575310e-06]
 [9.99998450e-01 1.49382470e-06 1.79321178e-14]
```

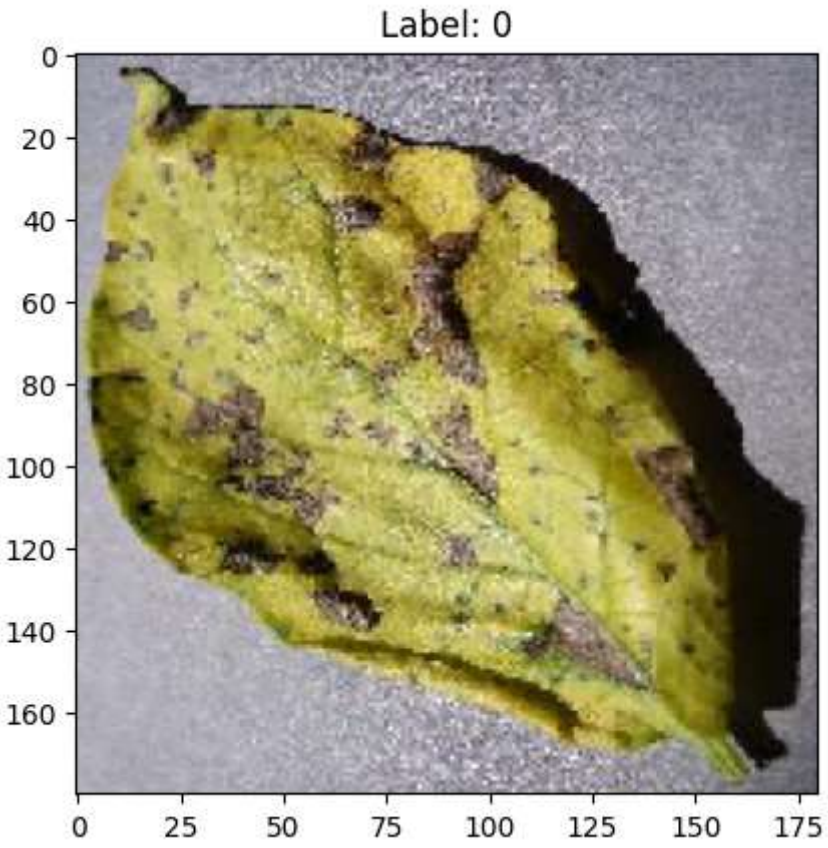
```
[9.99999762e-01 2.83465909e-07 1.10798848e-12]
[1.00000000e+00 1.69374834e-10 2.25314164e-21]
[3.58794965e-02 9.63740110e-01 3.80435522e-04]
[6.26137364e-07 9.99999046e-01 3.66806006e-07]
[1.00000000e+00 1.88241350e-12 2.33725698e-19]
[4.90213782e-01 4.99549091e-01 1.02371722e-02]
[9.98992741e-01 1.00710627e-03 8.47517470e-08]
[1.00000000e+00 3.14746096e-09 2.05734443e-16]
[1.18633604e-03 9.98802304e-01 1.13075284e-05]
[1.25153718e-04 9.99856830e-01 1.79497420e-05]
[1.48455547e-05 9.99984860e-01 3.38751619e-07]
[7.41504846e-06 9.99991655e-01 9.30060253e-07]
[9.99999642e-01 4.14586481e-07 5.38760059e-09]
[7.88095668e-02 9.20980573e-01 2.09888531e-04]
[1.19777750e-02 9.78441656e-01 9.58056096e-03]
[9.94390666e-01 5.60934795e-03 7.28447480e-09]
[6.76242280e-08 8.02716240e-03 9.91972685e-01]
[9.99999166e-01 8.52116159e-07 4.12111678e-10]
[1.92786320e-05 8.85973513e-01 1.14007168e-01]
[3.80859017e-01 6.19141042e-01 1.84498070e-10]
[1.00000000e+00 9.92707869e-13 2.63988695e-18]
[6.75656366e-08 9.99999881e-01 2.52840575e-08]
[7.61118606e-02 9.23707187e-01 1.80933450e-04]
[1.68439715e-06 9.99936938e-01 6.14092860e-05]
[4.68463952e-07 9.84864593e-01 1.51348850e-02]
[2.33820811e-01 7.66042411e-01 1.36768009e-04]
[1.87609912e-05 9.99981046e-01 1.85592782e-07]
[1.00000000e+00 3.33234718e-10 3.42422969e-16]
[1.00000000e+00 5.67699574e-14 3.81915805e-24]
[3.94544622e-06 5.09487763e-02 9.49047267e-01]
[1.00000000e+00 1.31959495e-11 2.09780190e-16]
[9.96020019e-01 3.97826871e-03 1.78977280e-06]
[9.91626978e-01 5.17344987e-03 3.19964625e-03]
[8.21377625e-07 9.99727786e-01 2.71413825e-04]
[6.11481084e-07 5.00435174e-01 4.99564230e-01]
[9.91157293e-01 8.56001303e-03 2.82725407e-04]
[9.99750793e-01 2.24930816e-04 2.43571103e-05]
[1.21641951e-05 9.99917984e-01 6.98031145e-05]
[1.00000000e+00 1.82069897e-11 3.56712825e-19]
[1.24072209e-02 9.86599565e-01 9.93177528e-04]
[1.89518978e-05 9.99981046e-01 1.93323424e-09]
[1.00000000e+00 1.67036246e-11 9.30812764e-16]
[1.00000000e+00 7.80577736e-09 7.97801963e-16]
[1.91508107e-08 9.99997020e-01 2.92970685e-06]
[1.00000000e+00 1.05743767e-12 1.01064354e-20]
[1.34532695e-06 9.99965429e-01 3.32132622e-05]
[8.91284784e-04 9.99075055e-01 3.36830235e-05]
[1.00000000e+00 6.08605050e-12 5.32293148e-22]
[1.00000000e+00 4.50432971e-08 1.27191653e-14]
[1.00000000e+00 2.43932370e-13 1.46261092e-21]
[3.26202530e-06 9.99926805e-01 6.99740267e-05]
[1.00000000e+00 1.29736293e-12 1.85082404e-23]
[6.08522157e-07 9.99997616e-01 1.81515657e-06]
[9.91569221e-01 8.42778664e-03 2.99461340e-06]
[1.00000000e+00 2.26300489e-09 3.77985361e-15]
[1.87260378e-03 9.98127401e-01 1.82688675e-09]
```

```
[7.69304006e-06 9.99988198e-01 4.10645407e-06]
[2.55078430e-06 9.99171138e-01 8.26394884e-04]
[9.99966860e-01 3.31920310e-05 2.55656456e-08]
[9.99942422e-01 5.76273596e-05 2.05364312e-08]
[1.28655401e-07 9.99988198e-01 1.16477713e-05]
[2.18155131e-01 7.81412005e-01 4.32852918e-04]
[8.53263438e-01 1.46735176e-01 1.30504657e-06]
[9.99998689e-01 1.35298365e-06 1.26604491e-11]
[9.99898314e-01 1.00609454e-04 1.09240523e-06]
[1.00000000e+00 2.16928255e-12 3.75016952e-21]
[9.99950290e-01 4.97643414e-05 3.42758022e-09]
[4.71265204e-02 9.52853858e-01 1.96447327e-05]
[9.99994040e-01 6.00747535e-06 1.64504261e-11]
[9.37054720e-05 9.98677552e-01 1.22876919e-03]
[6.84957377e-06 9.99979854e-01 1.33098765e-05]
[9.99999881e-01 7.51306288e-08 3.37066104e-11]
[9.99999762e-01 2.40302683e-07 4.34804130e-13]
[6.79347022e-06 9.99849200e-01 1.43983474e-04]
[1.00000000e+00 3.81493814e-15 5.22144509e-24]
[1.37099078e-05 9.99885321e-01 1.00948280e-04]
[8.90705360e-06 9.99976039e-01 1.50339019e-05]
[9.99999404e-01 5.50336665e-07 4.53716332e-15]
[9.81762767e-01 1.80458054e-02 1.91457977e-04]
[6.07966940e-05 9.99939203e-01 4.65700700e-10]
[4.40808117e-06 9.99995589e-01 1.88617904e-08]
[1.00000000e+00 2.16028774e-14 1.80915434e-26]
[2.05486183e-04 4.54034470e-02 9.54391062e-01]
[1.00000000e+00 4.52757248e-10 1.35746212e-18]
[2.40369127e-06 9.99963999e-01 3.35764016e-05]
[1.34438349e-04 9.99843717e-01 2.18160167e-05]
[8.07125807e-01 1.92270994e-01 6.03218446e-04]
[5.72216668e-05 9.99921441e-01 2.13303010e-05]
[9.84550655e-01 1.49571775e-02 4.92148043e-04]
[1.00000000e+00 6.07535723e-12 1.25377135e-19]
[8.15020394e-06 9.96883214e-01 3.10858688e-03]
[9.99999285e-01 7.53395568e-07 8.68641870e-09]
[9.51630235e-01 4.74807434e-02 8.89090938e-04]
[2.91100491e-06 9.99995112e-01 1.93245683e-06]
[1.00000000e+00 9.42029810e-15 7.94974287e-22]
[5.95867959e-07 9.99998450e-01 9.93530421e-07]
[1.87414959e-02 9.81253862e-01 4.61965783e-06]
[3.19085780e-06 9.99994516e-01 2.31222248e-06]
[2.88973912e-03 9.97105896e-01 4.30912542e-06]
[9.89941657e-01 1.00583266e-02 2.22021157e-09]
[1.00000000e+00 1.95917487e-08 2.36399585e-19]
[9.99997020e-01 2.96319240e-06 5.77631099e-09]
[1.00000000e+00 5.26141020e-10 3.57092878e-18]
[3.06119296e-06 9.99926567e-01 7.03272453e-05]
[1.00000000e+00 3.73080100e-09 1.01930944e-18]
[9.99975085e-01 2.43174272e-05 5.47437480e-07]
[1.00000000e+00 1.72030827e-11 6.43921964e-16]
[5.24546683e-01 4.75453347e-01 1.49408025e-12]
[1.40423595e-04 9.99859571e-01 1.36958334e-09]
[6.95692006e-06 9.99993086e-01 8.45780224e-10]
[5.53562073e-03 9.94462907e-01 1.41537396e-06]
[1.00000000e+00 2.05854171e-12 9.00687242e-20]
```

```
 [9.99562800e-01 4.37142822e-04 6.67663169e-09]
 [4.98561178e-07 9.99991298e-01 8.19327033e-06]
 [7.81610049e-03 9.92183626e-01 2.43105376e-07]
 [9.37973800e-06 9.99989390e-01 1.18858497e-06]
 [1.64793018e-05 9.99918461e-01 6.50830552e-05]
 [1.00000000e+00 8.51119952e-10 5.82381156e-16]
 [5.60443652e-07 6.70217931e-01 3.29781473e-01]
 [1.20513057e-02 9.84568000e-01 3.38065857e-03]
 [8.46183138e-06 9.99991536e-01 3.34392247e-08]
 [1.06356254e-06 9.99998927e-01 6.37999076e-09]
 [1.28491899e-06 9.99998569e-01 1.17047705e-07]
 [9.99997973e-01 2.06112713e-06 5.74343184e-10]
 [9.99773800e-01 2.26190969e-04 2.71029723e-14]
 [9.99999762e-01 1.88787581e-07 8.56359497e-11]
 [1.00000000e+00 2.22868848e-10 5.60162582e-15]
 [9.99998450e-01 1.58933108e-06 2.61440678e-11]
 [1.28095434e-03 9.63855207e-01 3.48638445e-02]
 [1.13853377e-04 9.59369063e-01 4.05171253e-02]
 [9.99974370e-01 2.56234562e-05 8.57339026e-12]
 [1.27021423e-07 9.98622894e-01 1.37701735e-03]
 [1.00000000e+00 3.15354831e-09 7.76942890e-17]
 [1.12582216e-04 9.99887109e-01 2.64488705e-07]
 [2.98427021e-05 2.27337837e-01 7.72632301e-01]
 [2.36924915e-08 1.04692066e-03 9.98953104e-01]
 [9.99989867e-01 1.01632213e-05 2.40683296e-09]
 [1.92712818e-03 9.98072863e-01 2.09946993e-08]
 [4.48100366e-07 9.99861121e-01 1.38466814e-04]
 [9.52923438e-06 9.94967401e-01 5.02307061e-03]
 [9.99946952e-01 5.30162943e-05 3.01244446e-10]
 [1.00000000e+00 9.55533153e-11 6.20979582e-19]
 [7.25649215e-07 9.99945402e-01 5.38711793e-05]
 [9.97271359e-01 2.72858329e-03 3.48872251e-11]
 [1.30575181e-05 9.09916341e-01 9.00706127e-02]
 [9.86278832e-01 1.32310875e-02 4.90101753e-04]
 [2.08279721e-06 9.99974966e-01 2.29231973e-05]
 [4.01598925e-07 9.99700546e-01 2.99057428e-04]
 [2.39892785e-07 9.99999642e-01 1.18464150e-07]
 [1.42695342e-07 9.33952570e-01 6.60473108e-02]
 [5.30064244e-06 5.70623338e-01 4.29371297e-01]
 [2.23832239e-06 1.56479403e-02 9.84349906e-01]
 [3.45037563e-07 1.75270170e-01 8.24729383e-01]
 [9.93139088e-01 6.86086854e-03 1.64671103e-08]
 [6.11105570e-05 9.99938607e-01 2.78295062e-07]
 [1.00000000e+00 8.59162630e-10 1.70638027e-15]
 [6.50079187e-07 9.99998212e-01 1.19173956e-06]
 [4.85590026e-05 9.99933958e-01 1.74265260e-05]
 [2.50416588e-06 9.99991775e-01 5.68560881e-06]
 [1.00000000e+00 2.16788752e-17 1.26178398e-27]]
[1 0 1 0 1 0 1 1 1 1 1 1 1 2 1 1 0 0 0 1 0 2 0 0 1 0 1 1 1 0 1 2 1 1 1 1 1
 0 1 1 0 0 0 0 1 0 0 1 2 1 1 0 0 1 0 0 0 1 1 0 1 0 0 1 1 1 1 0 1 1 0 2 0 1
 1 0 1 1 1 1 1 1 0 0 2 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 1 0 1 0 0 1
 1 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 2 0 1 1 0 1 0 0 1 0 0
 1 0 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1
 0 1 0 1 2 2 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 2 2 0 1 0 1 1 1 0]
```

In [58]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
datagen_test = ImageDataGenerator(rescale=1./255)
test_generator = datagen_test.flow_from_dataframe(
    dataframe=test_df,
    directory=None,
    x_col='image_path',
    y_col='label',
    target_size=(180, 180),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
images_batch, labels_batch = next(test_generator)
first_image = images_batch[0]
first_label = labels_batch[0]

print(f"First image shape: {first_image.shape}")
print(f"First label: {first_label}")
import matplotlib.pyplot as plt

plt.imshow(first_image)
plt.title(f"Label: {np.argmax(first_label)}")
plt.show()
```

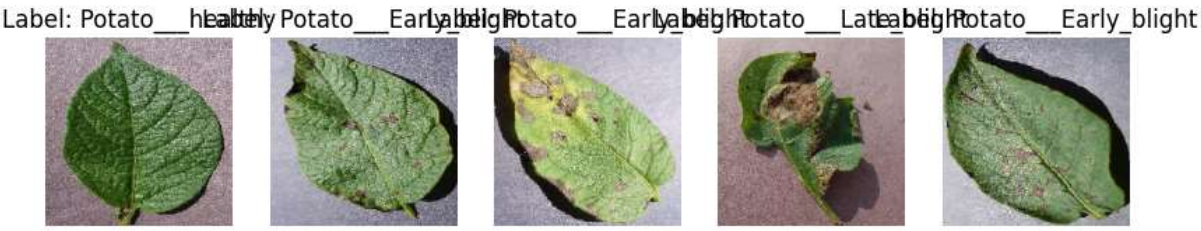
```
Found 216 validated image filenames belonging to 3 classes.
First image shape: (180, 180, 3)
First label: [1. 0. 0.]
```



In [70]:
```
import matplotlib.pyplot as plt
```

```
def display_batch(generator, num_images=5):

    images_batch, labels_batch = next(generator)


    num_images = min(num_images, len(images_batch))


    plt.figure(figsize=(10, 10))


    for i in range(num_images):
        plt.subplot(1, num_images, i + 1)
        plt.imshow(images_batch[i])
        plt.title(f"Label: {np.argmax(labels_batch[i])}")
        plt.axis('off')

    plt.show()
display_batch(test_generator, num_images=5)
```

In [61]:
```
import matplotlib.pyplot as plt
import numpy as np
class_labels = list(train_generator.class_indices.keys())
def display_batch_with_labels(generator, num_images=5):
    images_batch, labels_batch = next(generator)


    num_images = min(num_images, len(images_batch))


    plt.figure(figsize=(10, 10))


    for i in range(num_images):
        plt.subplot(1, num_images, i + 1)
        plt.imshow(images_batch[i])
        plt.title(f"Label: {class_labels[np.argmax(labels_batch[i])]}")  # Use clas
        plt.axis('off')

    plt.show()


display_batch_with_labels(test_generator, num_images=5)
```



In [62]:
```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
```

In [65]:
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf


class_names = list(train_generator.class_indices.keys())


def predict(model, image):
    # Preprocess image if needed
    image = np.expand_dims(image, axis=0)
    predictions = model.predict(image)
    predicted_class = np.argmax(predictions[0])
    confidence = np.max(predictions[0])
    return class_names[predicted_class], confidence


plt.figure(figsize=(15, 15))
for images_batch, labels_batch in test_generator:
    for i in range(min(9, len(images_batch))):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images_batch[i])


        predicted_class, confidence = predict(model, images_batch[i])

        actual_class = class_names[np.argmax(labels_batch[i])]


        plt.title(f"Actual: {actual_class},\nPredicted: {predicted_class},\nConfide
        plt.axis("off")

    plt.show()
    break
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 300ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
```

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 0.93

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 1.00

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 1.00

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 1.00

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 1.00

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 1.00

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 1.00

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 1.00

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 1.00

In [ ]: