In [1]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
from PIL import Image

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adamax
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img, img_to_a
from tensorflow.keras.callbacks import EarlyStopping

import warnings
warnings.filterwarnings('ignore')
```

Data Processing

In [2]:
```
train_dir = '/kaggle/input/brain-tumor-mri-dataset/Training'

classes = os.listdir(train_dir)
classes
```

Out[2]:
```
['pituitary', 'notumor', 'meningioma', 'glioma']
```

In [3]:
```
image_paths = []
image_labels = []

categories = os.listdir(train_dir)

for category in categories:
    category_path = os.path.join(train_dir, category)
    images = os.listdir(category_path)

    for image in images:
        image_path = os.path.join(category_path, image)
        image_paths.append(image_path)
        image_labels.append(category)

train_df = pd.DataFrame(data={'filepaths': image_paths, 'labels': image_labels})
train_df
```

Out[3]:

	filepaths	labels
0	/kaggle/input/brain-tumor-mri-dataset/Training...	pituitary
1	/kaggle/input/brain-tumor-mri-dataset/Training...	pituitary
2	/kaggle/input/brain-tumor-mri-dataset/Training...	pituitary
3	/kaggle/input/brain-tumor-mri-dataset/Training...	pituitary
4	/kaggle/input/brain-tumor-mri-dataset/Training...	pituitary
...	...	...
5707	/kaggle/input/brain-tumor-mri-dataset/Training...	glioma
5708	/kaggle/input/brain-tumor-mri-dataset/Training...	glioma
5709	/kaggle/input/brain-tumor-mri-dataset/Training...	glioma
5710	/kaggle/input/brain-tumor-mri-dataset/Training...	glioma
5711	/kaggle/input/brain-tumor-mri-dataset/Training...	glioma

5712 rows × 2 columns

In [4]:

```
test_dir ='/kaggle/input/brain-tumor-mri-dataset/Testing'

classes = os.listdir(test_dir)
classes
```

Out[4]:

```
['pituitary', 'notumor', 'meningioma', 'glioma']
```

In [5]:

```
image_paths = []
image_labels = []

categories = os.listdir(test_dir)

for category in categories:
    category_path = os.path.join(test_dir, category)
    images = os.listdir(category_path)

    for image in images:
        image_path = os.path.join(category_path, image)
        image_paths.append(image_path)
        image_labels.append(category)

test_df = pd.DataFrame(data={'filepaths': image_paths, 'labels': image_labels})
test_df
```

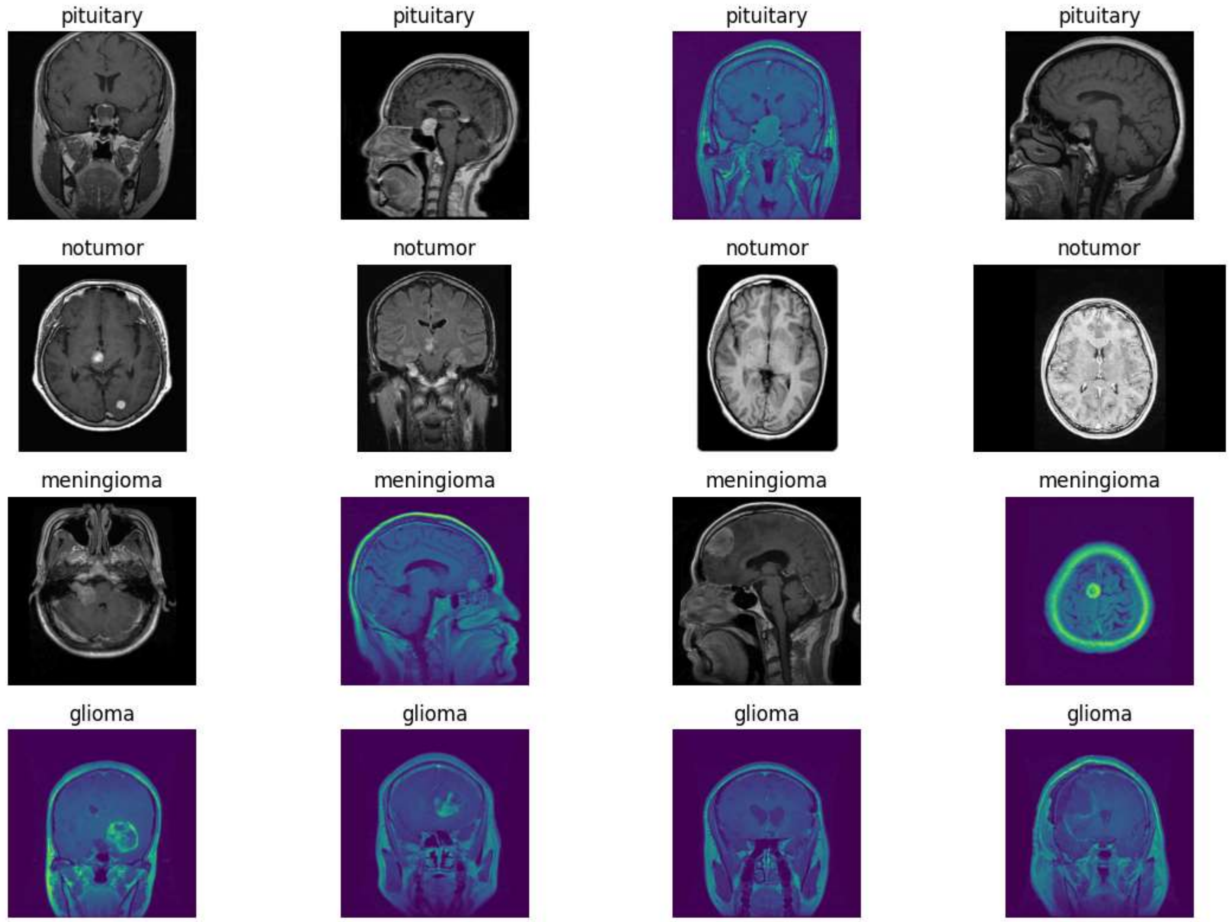
Out[5]:

	filepaths	labels
0	/kaggle/input/brain-tumor-mri-dataset/Testing/...	pituitary
1	/kaggle/input/brain-tumor-mri-dataset/Testing/...	pituitary
2	/kaggle/input/brain-tumor-mri-dataset/Testing/...	pituitary
3	/kaggle/input/brain-tumor-mri-dataset/Testing/...	pituitary
4	/kaggle/input/brain-tumor-mri-dataset/Testing/...	pituitary
...	...	...
1306	/kaggle/input/brain-tumor-mri-dataset/Testing/...	glioma
1307	/kaggle/input/brain-tumor-mri-dataset/Testing/...	glioma
1308	/kaggle/input/brain-tumor-mri-dataset/Testing/...	glioma
1309	/kaggle/input/brain-tumor-mri-dataset/Testing/...	glioma
1310	/kaggle/input/brain-tumor-mri-dataset/Testing/...	glioma

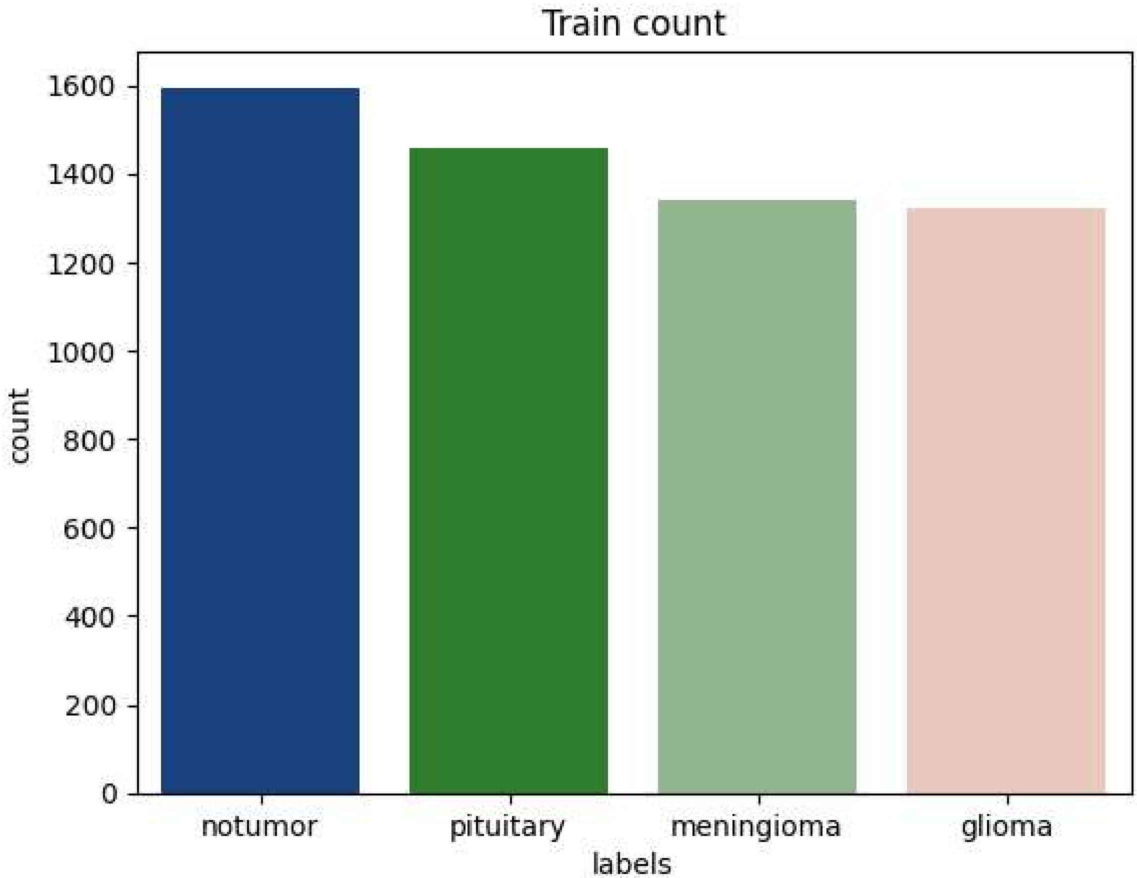
1311 rows × 2 columns

In [6]:

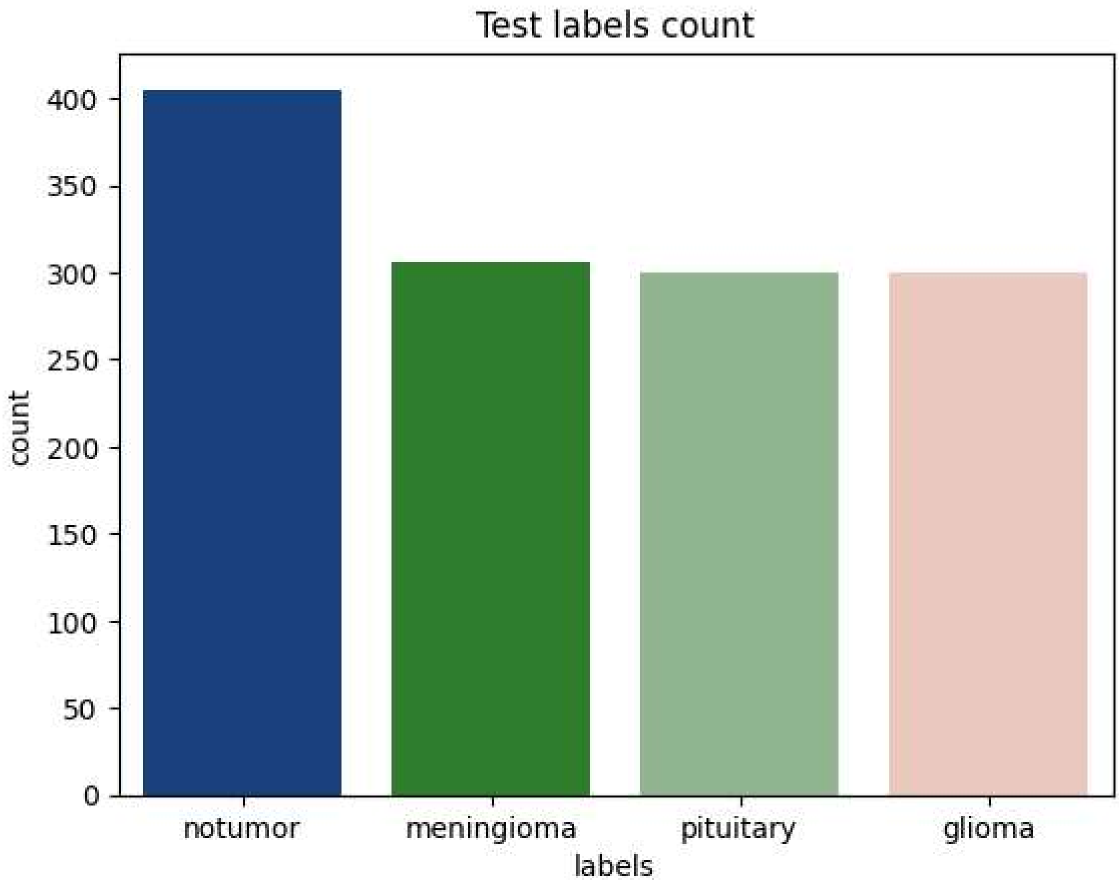
```
def plot_class_samples(df, classes, num_samples=4):
    plt.figure(figsize=(12,8))

    for i, cls in enumerate(classes):
        class_images = df[df['labels'] == cls]['filepaths'].sample(num_samples, random

        for j, img_path in enumerate(class_images):
            img = Image.open(img_path)  # Load the image
            plt.subplot(len(classes), num_samples, i * num_samples + j + 1)
            plt.imshow(img)
            plt.axis('off')
            plt.title(cls)

    plt.tight_layout()
    plt.show()
classes = train_df['labels'].unique()

plot_class_samples(train_df, classes)
```



In [11]:
```
custom_palette = ["#0B3D91", "#228B22", "#8FBC8F", "#F2C3B9", "#556B2F", "#2F4f4f"]
sns.countplot(data=train_df, x='labels', palette=custom_palette, order=train_df['label
plt.title('Train count')
plt.show()
```



In [13]:
```
sns.countplot(data=test_df,x='labels',palette=custom_palette,order=test_df['labels'].v
plt.title('Test labels count')
plt.show()
```



CNN Model and Splitting

In [14]:
```
train_df.shape[0]
```
Out[14]:
```
5712
```

In [15]:
```
train_df, valid_df = train_test_split(train_df, test_size=0.2, random_state=42)
```

In [16]:
```
valid_df.shape[0]
```
Out[16]:
```
1143
```

In [17]:
```
train_df.shape[0]
```
Out[17]:
```
4569
```

In [18]:
```
image_gen = ImageDataGenerator(rescale=1/255)
```

In [19]:
```
gen_train=image_gen.flow_from_dataframe(train_df,x_col='filepaths',y_col='labels',targ
gen_valid=image_gen.flow_from_dataframe(valid_df,x_col='filepaths',y_col='labels',targ
gen_test=image_gen.flow_from_dataframe(test_df,x_col='filepaths',y_col='labels',target
```
```
Found 4569 validated image filenames belonging to 4 classes.
Found 1143 validated image filenames belonging to 4 classes.
Found 1311 validated image filenames belonging to 4 classes.
```

In [20]:
```
model=Sequential()

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(224,224,3), activation='re
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(256,activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(4,activation='softmax'))

model.compile(optimizer=Adamax(learning_rate=0.001),loss='categorical_crossentropy',me
```

In [21]:
```
early_stopping=EarlyStopping(monitor='val_loss',mode='min',patience=7)
```

In [22]:
```
history= model.fit(gen_train,validation_data=gen_valid,epochs=30,callbacks=[early_stop
```

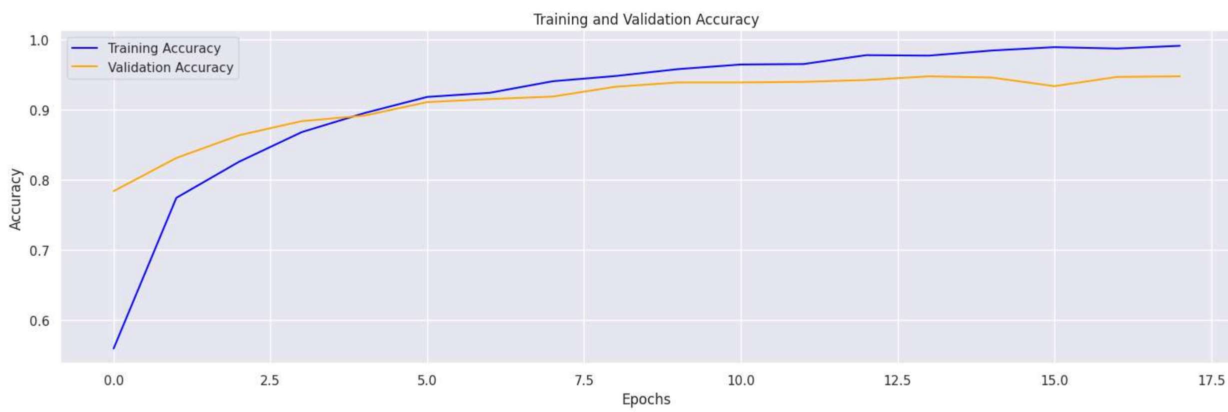
```
Epoch 1/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 52s 266ms/step - accuracy: 0.4516 - loss: 1.1561 - val_a
ccuracy: 0.7839 - val_loss: 0.6118
Epoch 2/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 141ms/step - accuracy: 0.7577 - loss: 0.6077 - val_a
ccuracy: 0.8311 - val_loss: 0.4575
Epoch 3/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 140ms/step - accuracy: 0.8253 - loss: 0.4536 - val_a
ccuracy: 0.8635 - val_loss: 0.3696
Epoch 4/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.8684 - loss: 0.3416 - val_a
ccuracy: 0.8836 - val_loss: 0.3351
Epoch 5/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.8853 - loss: 0.2989 - val_a
ccuracy: 0.8915 - val_loss: 0.3056
Epoch 6/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9110 - loss: 0.2398 - val_a
ccuracy: 0.9108 - val_loss: 0.2650
Epoch 7/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 143ms/step - accuracy: 0.9294 - loss: 0.2063 - val_a
ccuracy: 0.9151 - val_loss: 0.2422
Epoch 8/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9406 - loss: 0.1738 - val_a
ccuracy: 0.9186 - val_loss: 0.2251
Epoch 9/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9503 - loss: 0.1402 - val_a
ccuracy: 0.9326 - val_loss: 0.2128
Epoch 10/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 143ms/step - accuracy: 0.9551 - loss: 0.1163 - val_a
ccuracy: 0.9388 - val_loss: 0.2128
Epoch 11/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 143ms/step - accuracy: 0.9637 - loss: 0.1234 - val_a
ccuracy: 0.9388 - val_loss: 0.1963
Epoch 12/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 143ms/step - accuracy: 0.9674 - loss: 0.0891 - val_a
ccuracy: 0.9396 - val_loss: 0.2103
Epoch 13/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9762 - loss: 0.0678 - val_a
ccuracy: 0.9423 - val_loss: 0.1984
Epoch 14/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 141ms/step - accuracy: 0.9795 - loss: 0.0658 - val_a
ccuracy: 0.9475 - val_loss: 0.2009
Epoch 15/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 145ms/step - accuracy: 0.9867 - loss: 0.0440 - val_a
ccuracy: 0.9458 - val_loss: 0.2344
Epoch 16/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9886 - loss: 0.0390 - val_a
ccuracy: 0.9335 - val_loss: 0.3254
Epoch 17/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 141ms/step - accuracy: 0.9811 - loss: 0.0551 - val_a
ccuracy: 0.9466 - val_loss: 0.2395
Epoch 18/30
143/143 ━━━━━━━━━━━━━━━━━━━━ 21s 142ms/step - accuracy: 0.9925 - loss: 0.0228 - val_a
ccuracy: 0.9475 - val_loss: 0.2486
```

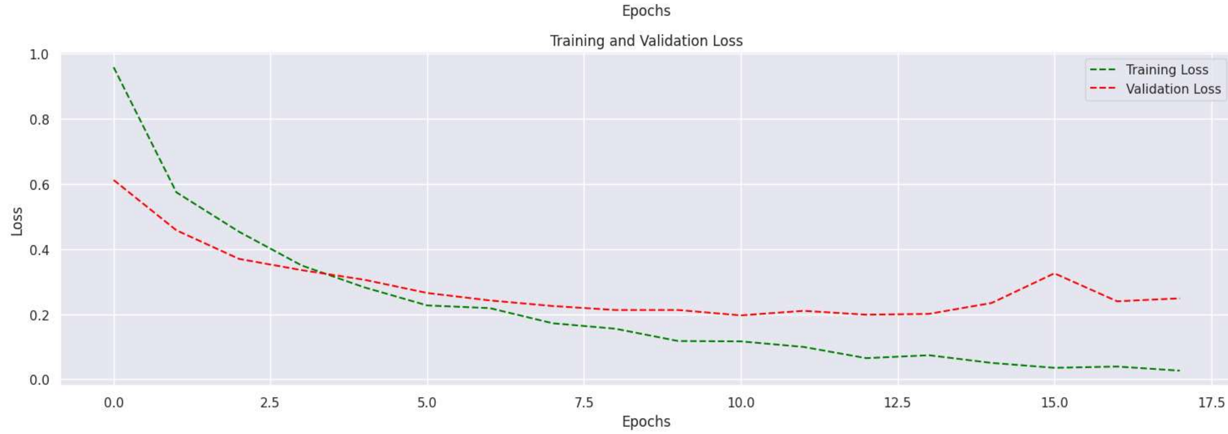
In [24]:

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, c
def plot_loss_accuracy(history, figsize=(15, 10)):
    sns.set() # Use seaborn styling for better aesthetics
    # Create a figure with two subplots (2 rows, 1 column)
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=figsize)
```

```
    # Plot Training and Validation Accuracy on the first subplot
    ax1.plot(history.epoch, history.history["accuracy"], label='Training Accuracy', co
    ax1.plot(history.epoch, history.history["val_accuracy"], label='Validation Accurac
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Accuracy')
    ax1.legend()
    ax1.set_title('Training and Validation Accuracy')
    # Plot Training and Validation Loss on the second subplot
    ax2.plot(history.epoch, history.history["loss"], label='Training Loss', color='gre
    ax2.plot(history.epoch, history.history["val_loss"], label='Validation Loss', colo
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Loss')
    ax2.legend()
    ax2.set_title('Training and Validation Loss')
    # Adjust layout for better spacing
    plt.tight_layout()
    # Display the plot
    plt.show()
```

In [25]:
```
#Plot training history
plot_loss_accuracy(history)
```





In [26]:
```
pred=model.predict(gen_test)
predictions=np.argmax(pred,axis=1)
print(classification_report(gen_test.classes,predictions))
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 7s 176ms/step
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       300
           1       0.92      0.91      0.91       306
           2       0.98      1.00      0.99       405
           3       0.97      1.00      0.99       300

    accuracy                           0.96      1311
   macro avg       0.95      0.95      0.95      1311
weighted avg       0.96      0.96      0.96      1311
```

In [27]:
```
model.evaluate(gen_test)
```

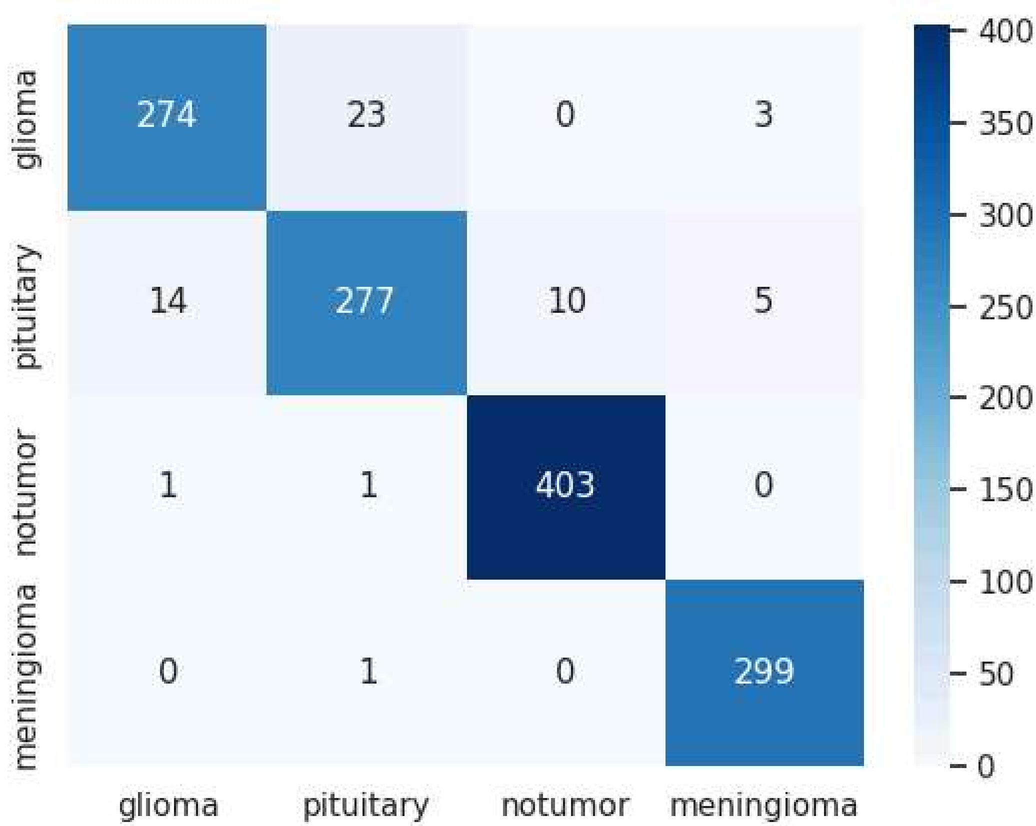
```
41/41 ━━━━━━━━━━━━━━━━━━━━ 4s 91ms/step - accuracy: 0.9825 - loss: 0.0736
```

Out[27]:
```
[0.18101365864276886, 0.9557589888572693]
```

In [35]:
```
model.save('brain_tumor_model.h5')
```

In [31]:
```
conf_matrix = confusion_matrix(gen_test.classes, predictions)
labels = list(train_df['labels'].unique())
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=labels, ytickl
```

Out[31]:
```
<Axes: >
```



Prediction on Random Images

In [32]:
```
import random
random_indices = random.sample(range(len(test_df)), 4)
```

```python
random_paths = test_df.iloc[random_indices]['filepaths'].values
actual_labels = test_df.iloc[random_indices]['labels'].values

plt.figure(figsize=(16, 4))

for i, img_path in enumerate(random_paths):
    img = load_img(img_path, target_size=(224, 224))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0

    pred = model.predict(img_array)
    predicted_class = np.argmax(pred, axis=1)[0]
    class_labels = list(gen_train.class_indices.keys())
    predicted_label = class_labels[predicted_class]

    plt.subplot(1, 4, i + 1)
    plt.imshow(img)
    plt.axis('off')
    plt.title(f'Actual: {actual_labels[i]}\nPredicted: {predicted_label}')

plt.tight_layout()
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 742ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
```