In [3]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\aiswarya\anaconda3\lib
\site-packages (2.18.0)
Requirement already satisfied: tensorflow-intel==2.18.0 in c:\users\aiswarya
\anaconda3\lib\site-packages (from tensorflow) (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\aiswarya\anacond
a3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\aiswarya\anac
onda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (24.12.2
3)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\user
s\aiswarya\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensor
flow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\aiswarya\anaconda
3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\aiswarya\anacond
a3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\aiswarya\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.18.0->tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=
4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\aiswarya\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.18.0->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\aiswarya\anaconda3\lib
\site-packages (from tensorflow-intel==2.18.0->tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\aiswarya\anaconda3\lib
\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\aiswarya\anaconda
3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\aiswarya
\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.1
1.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\aiswarya\anaconda3\l
ib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.69.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in c:\users\aiswarya\a
naconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.18.
0)
Requirement already satisfied: keras>=3.5.0 in c:\users\aiswarya\anaconda3\li
b\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in c:\users\aiswarya\anac
onda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\users\aiswarya\anaconda3\li
b\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.11.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in c:\users\aiswarya\a
naconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.4.
1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\aiswarya\anacon
da3\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.18.0->tens
orflow) (0.44.0)
Requirement already satisfied: rich in c:\users\aiswarya\anaconda3\lib\site-p
```

```
ackages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\aiswarya\anaconda3\lib\site-
packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\aiswarya\anaconda3\lib\site
-packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.13.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\aiswarya
\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.1
8.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\aiswarya\anaconda3\li
b\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorfl
ow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\aiswarya\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->te
nsorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\aiswarya\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->te
nsorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->t
ensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\us
ers\aiswarya\anaconda3\lib\site-packages (from tensorboard<2.19,>=2.18->tenso
rflow-intel==2.18.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->t
ensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\aiswarya\anacond
a3\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorfl
ow-intel==2.18.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\aiswarya\ana
conda3\lib\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->
tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\aiswarya\a
naconda3\lib\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0
->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\aiswarya\anaconda3\lib
\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow-in
tel==2.18.0->tensorflow) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\aiswarya\anaconda3\lib
\site-packages (2.18.0)
Requirement already satisfied: tensorflow-intel==2.18.0 in c:\users\aiswarya
\anaconda3\lib\site-packages (from tensorflow) (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\aiswarya\anacond
a3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\aiswarya\anac
onda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (24.12.2
3)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\user
s\aiswarya\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensor
flow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\aiswarya\anaconda
3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\aiswarya\anacond
a3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\aiswarya\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.18.0->tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=
4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\aiswarya\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.18.0->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\aiswarya\anaconda3\lib
\site-packages (from tensorflow-intel==2.18.0->tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\aiswarya\anaconda3\lib
\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\aiswarya\anaconda
3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\aiswarya
\anaconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.1
1.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\aiswarya\anaconda3\l
ib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\aiswarya\anaco
nda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.69.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in c:\users\aiswarya\a
naconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.18.
0)
Requirement already satisfied: keras>=3.5.0 in c:\users\aiswarya\anaconda3\li
b\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in c:\users\aiswarya\anac
onda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\users\aiswarya\anaconda3\li
b\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.11.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in c:\users\aiswarya\a
naconda3\lib\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.4.
1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\aiswarya\anacon
da3\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.18.0->tens
orflow) (0.44.0)
Requirement already satisfied: rich in c:\users\aiswarya\anaconda3\lib\site-p
```

```
ackages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\aiswarya\anaconda3\lib\site-
packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\aiswarya\anaconda3\lib\site
-packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.13.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\aiswarya
\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.1
8.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\aiswarya\anaconda3\li
b\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorfl
ow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\aiswarya\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->te
nsorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\aiswarya\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->te
nsorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->t
ensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\us
ers\aiswarya\anaconda3\lib\site-packages (from tensorboard<2.19,>=2.18->tenso
rflow-intel==2.18.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\aiswarya\anaconda3
\lib\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->t
ensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\aiswarya\anacond
a3\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorfl
ow-intel==2.18.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\aiswarya\ana
conda3\lib\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->
tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\aiswarya\a
naconda3\lib\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0
->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\aiswarya\anaconda3\lib
\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow-in
tel==2.18.0->tensorflow) (0.1.0)
```

In [7]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalMaxPooling2D,GlobalAveragePooling2
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, Ten
from sklearn.metrics import classification_report,confusion_matrix
from tqdm import tqdm
```

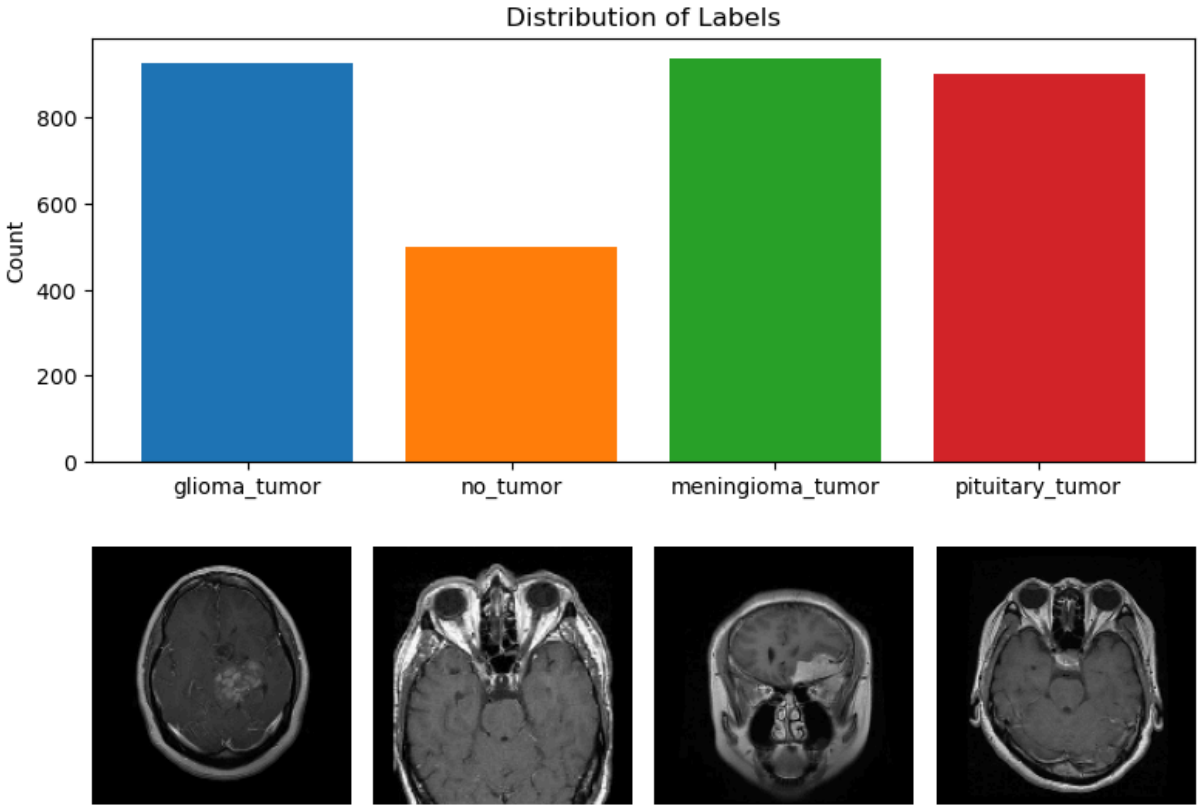
In [9]:
```
X_train = []
y_train = []
```

```
labels = ['glioma_tumor','no_tumor','meningioma_tumor','pituitary_tumor']


image_size = 150
for i in labels:
    folderPath = os.path.join(r"C:\Users\AISWARYA\Documents\Training",i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join(r"C:\Users\AISWARYA\Documents\Testing",i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

```
100%|██████████| 826/826 [00:03<00:00, 228.67it/s]
100%|██████████| 395/395 [00:01<00:00, 250.17it/s]
100%|██████████| 822/822 [00:03<00:00, 217.20it/s]
100%|██████████| 827/827 [00:04<00:00, 189.66it/s]
100%|██████████| 100/100 [00:00<00:00, 222.07it/s]
100%|██████████| 105/105 [00:00<00:00, 319.66it/s]
100%|██████████| 115/115 [00:00<00:00, 263.54it/s]
100%|██████████| 74/74 [00:00<00:00, 102.55it/s]
```

In [11]:
```
label_counts = {label: np.sum(y_train == label) for label in labels}

plt.figure(figsize=(8, 6))

colors = ["C0", "C1", "C2", "C3"]

plt.subplot(2, 1, 1)
bars = plt.bar(label_counts.keys(), label_counts.values(), color=colors)
# plt.xlabel('Labels')
plt.ylabel('Count')
plt.title('Distribution of Labels')

# Plot sample images from each label
k = 0
for i in labels:
    j = 0
    while True:
        if y_train[j] == i:
            plt.subplot(2, 4, k + 5)
            plt.imshow(X_train[j])
            plt.axis('off')
            k += 1
            break
        j += 1
```

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



In [13]:
```
X_train, y_train = shuffle(X_train,y_train, random_state=42)
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train, test_size=
```

In [15]:
```
y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)

y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

In [17]:
```
y_test[0]
```

Out[17]:
```
array([0., 0., 0., 1.])
```

In [19]:
```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

In [21]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dr
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, mi

model.summary()
```

```
C:\Users\AISWARYA\anaconda3\Lib\site-packages\keras\src\layers\convolutional
\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argu
ment 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)
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Pa
conv2d (Conv2D)	(None, 148, 148, 32)	
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	
conv2d_1 (Conv2D)	(None, 72, 72, 64)	1
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	
conv2d_2 (Conv2D)	(None, 34, 34, 128)	7
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	
conv2d_3 (Conv2D)	(None, 15, 15, 256)	29
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 256)	
flatten (Flatten)	(None, 12544)	
dense (Dense)	(None, 256)	3,21
dropout (Dropout)	(None, 256)	
dense_1 (Dense)	(None, 4)	

Total params: 3,600,964 (13.74 MB)

Trainable params: 3,600,964 (13.74 MB)

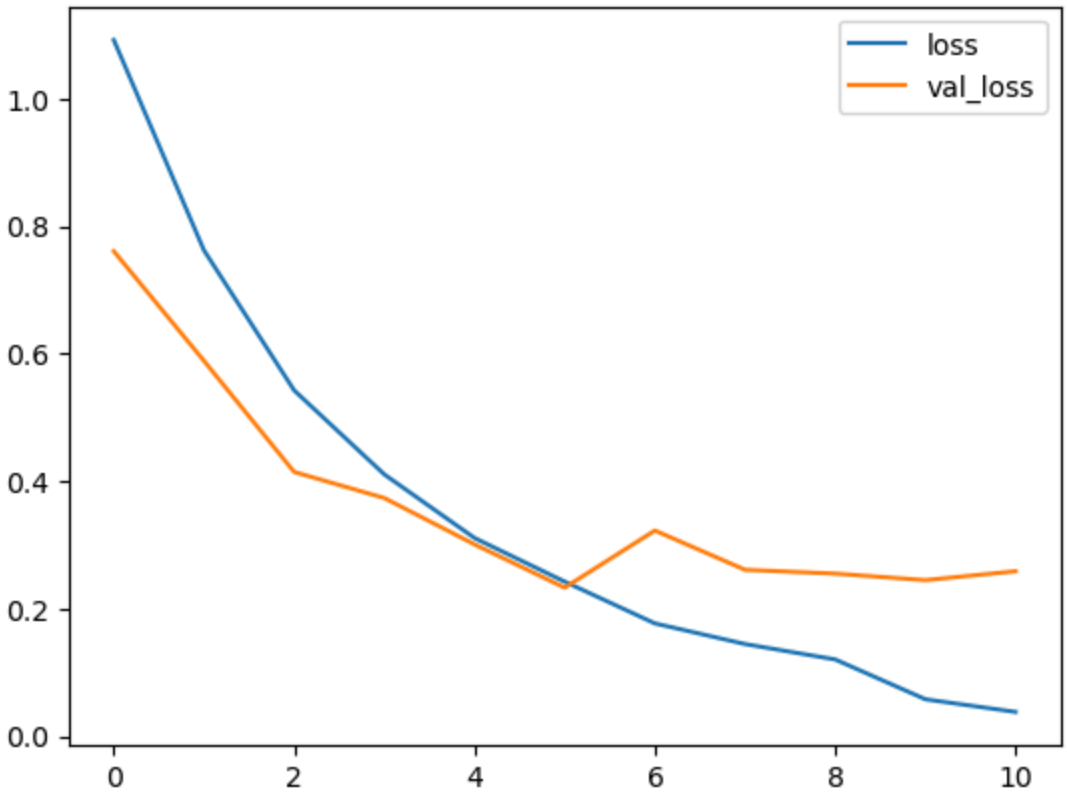
Non-trainable params: 0 (0.00 B)

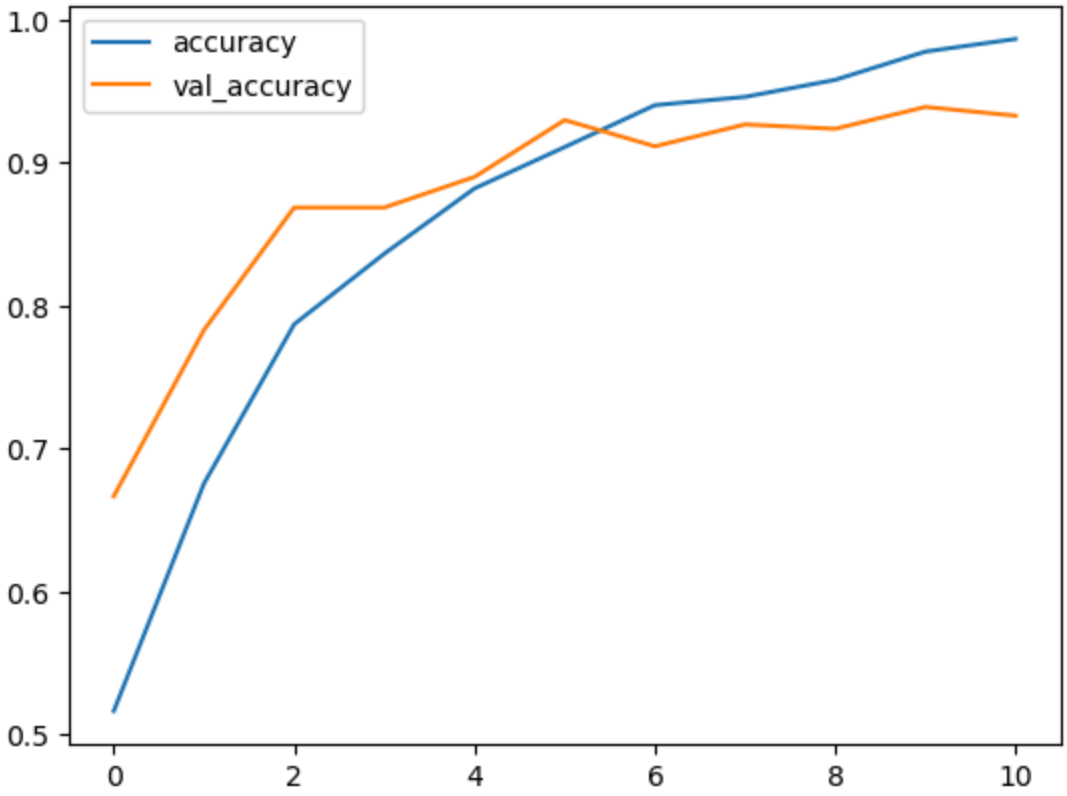
In [23]:
```
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=50,
    batch_size=16,
    callbacks=[early_stopping, reduce_lr]
)
```

```
Epoch 1/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 136s 713ms/step - accuracy: 0.4430 - loss:
1.2374 - val_accuracy: 0.6667 - val_loss: 0.7606 - learning_rate: 0.0010
Epoch 2/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 131s 711ms/step - accuracy: 0.6419 - loss:
0.8274 - val_accuracy: 0.7829 - val_loss: 0.5885 - learning_rate: 0.0010
Epoch 3/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 131s 712ms/step - accuracy: 0.7736 - loss:
0.5603 - val_accuracy: 0.8685 - val_loss: 0.4142 - learning_rate: 0.0010
Epoch 4/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 140s 698ms/step - accuracy: 0.8240 - loss:
0.4288 - val_accuracy: 0.8685 - val_loss: 0.3733 - learning_rate: 0.0010
Epoch 5/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 127s 689ms/step - accuracy: 0.8825 - loss:
0.3074 - val_accuracy: 0.8899 - val_loss: 0.3006 - learning_rate: 0.0010
Epoch 6/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 128s 697ms/step - accuracy: 0.9064 - loss:
0.2597 - val_accuracy: 0.9297 - val_loss: 0.2328 - learning_rate: 0.0010
Epoch 7/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 144s 705ms/step - accuracy: 0.9440 - loss:
0.1626 - val_accuracy: 0.9113 - val_loss: 0.3224 - learning_rate: 0.0010
Epoch 8/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 125s 678ms/step - accuracy: 0.9459 - loss:
0.1389 - val_accuracy: 0.9266 - val_loss: 0.2608 - learning_rate: 0.0010
Epoch 9/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 117s 634ms/step - accuracy: 0.9593 - loss:
0.1181 - val_accuracy: 0.9235 - val_loss: 0.2547 - learning_rate: 0.0010
Epoch 10/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 125s 677ms/step - accuracy: 0.9740 - loss:
0.0674 - val_accuracy: 0.9388 - val_loss: 0.2445 - learning_rate: 2.0000e-04
Epoch 11/50
184/184 ━━━━━━━━━━━━━━━━━━━━ 124s 672ms/step - accuracy: 0.9849 - loss:
0.0415 - val_accuracy: 0.9327 - val_loss: 0.2585 - learning_rate: 2.0000e-04
```

In [25]:
```
import pandas as pd
ef=pd.DataFrame(history.history)
ef[['loss','val_loss']].plot()
ef[['accuracy','val_accuracy']].plot()
```

Out[25]:
```
<Axes: >
```



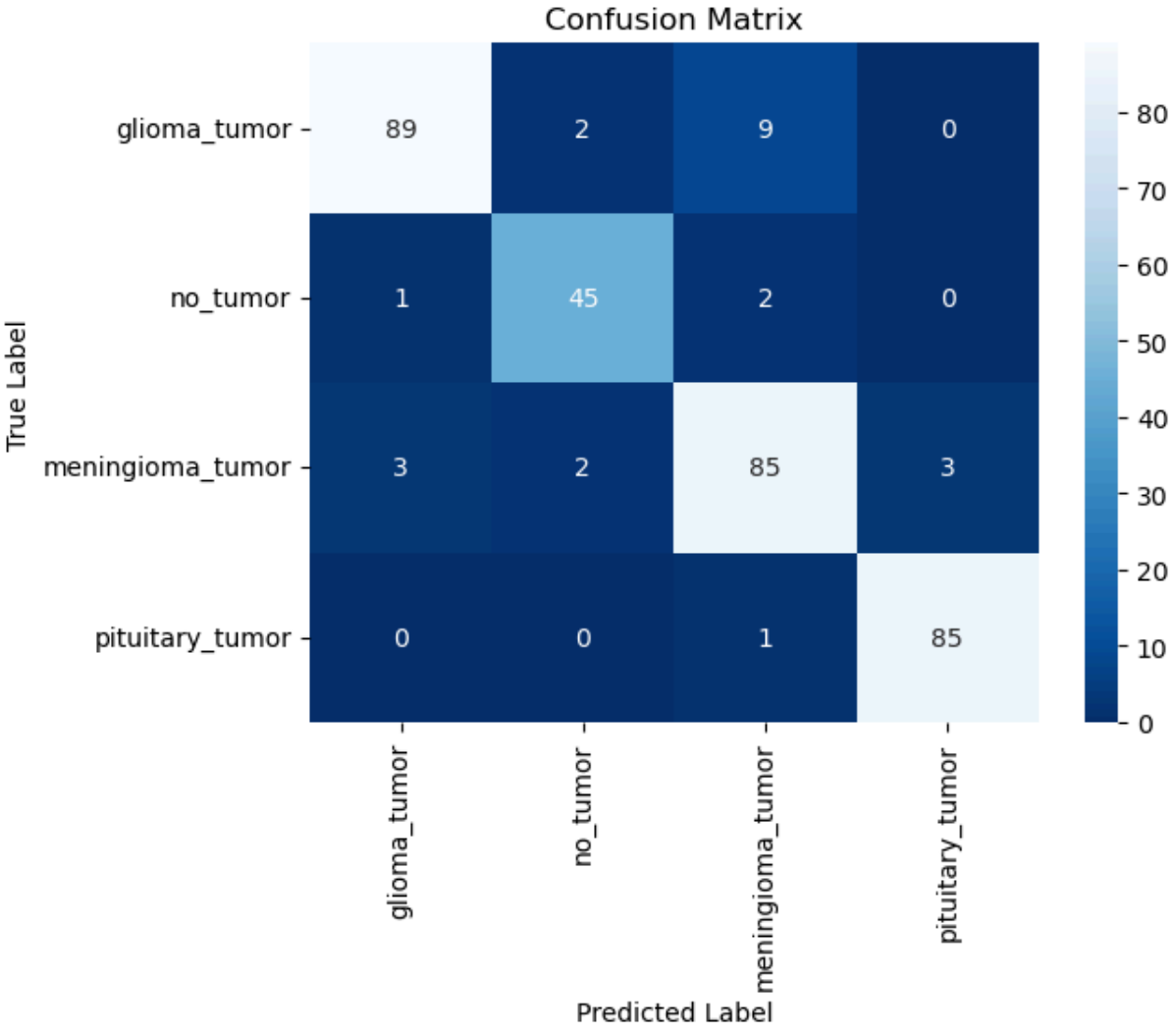


In [27]:
```
loss,test=model.evaluate(X_test,y_test)
print("loss : ",loss)
print("accuracy : ",test)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 4s 321ms/step - accuracy: 0.9317 - loss: 0.196
8
loss :  0.23275290429592133
accuracy :  0.9296635985374451
```

In [29]:
```
y_true_test = np.argmax(y_test, axis=1)
y_pred_test = np.argmax(model.predict(X_test), axis=1)

heatmap = sns.heatmap(confusion_matrix(y_true_test,y_pred_test), annot=True,
                      xticklabels=labels, yticklabels=labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 4s 376ms/step
```



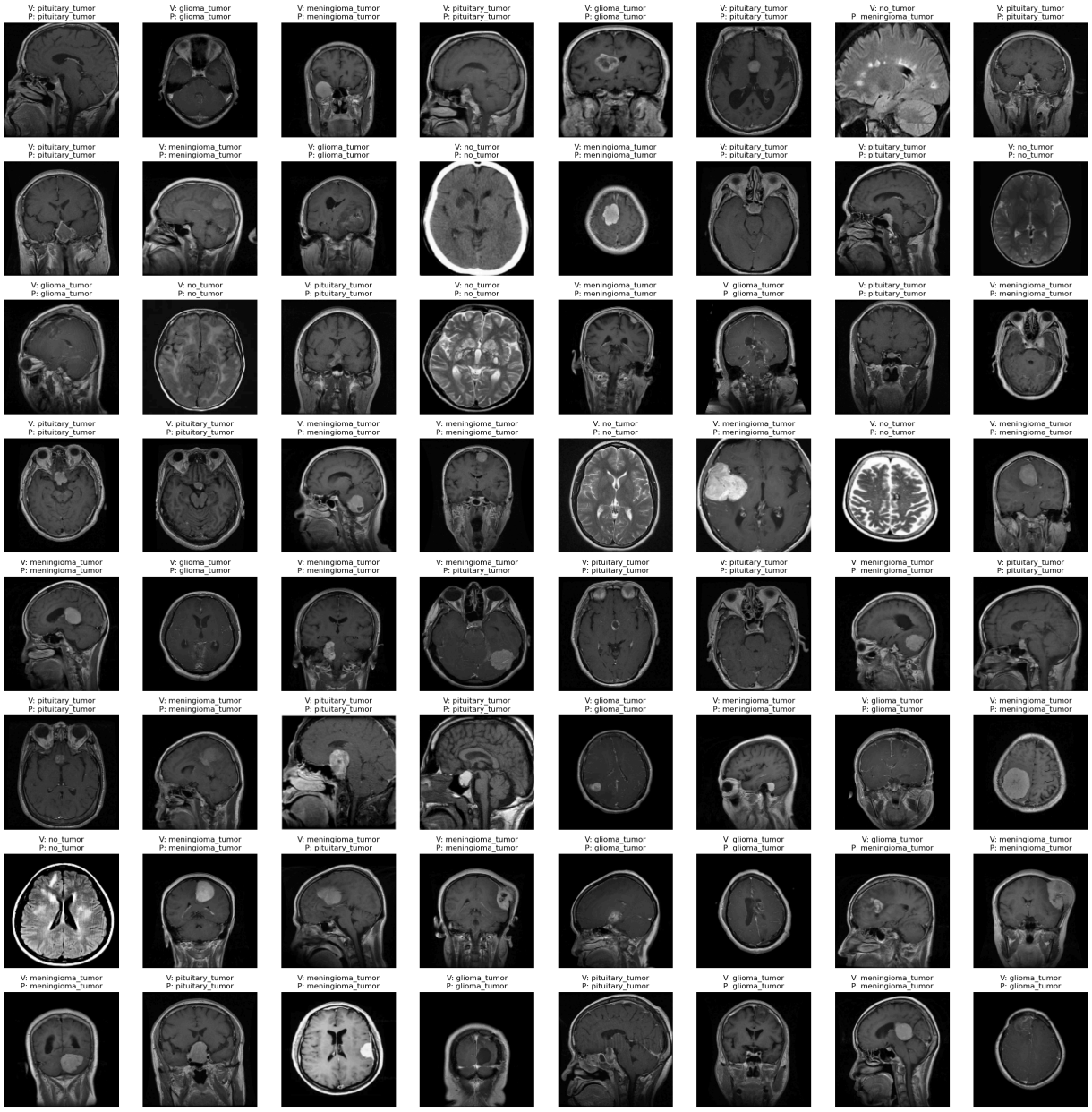
In [31]:
```
print(classification_report(y_true_test,y_pred_test))
```

	precision	recall	f1-score	support
0	0.96	0.89	0.92	100
1	0.92	0.94	0.93	48
2	0.88	0.91	0.89	93
3	0.97	0.99	0.98	86
accuracy			0.93	327
macro avg	0.93	0.93	0.93	327
weighted avg	0.93	0.93	0.93	327

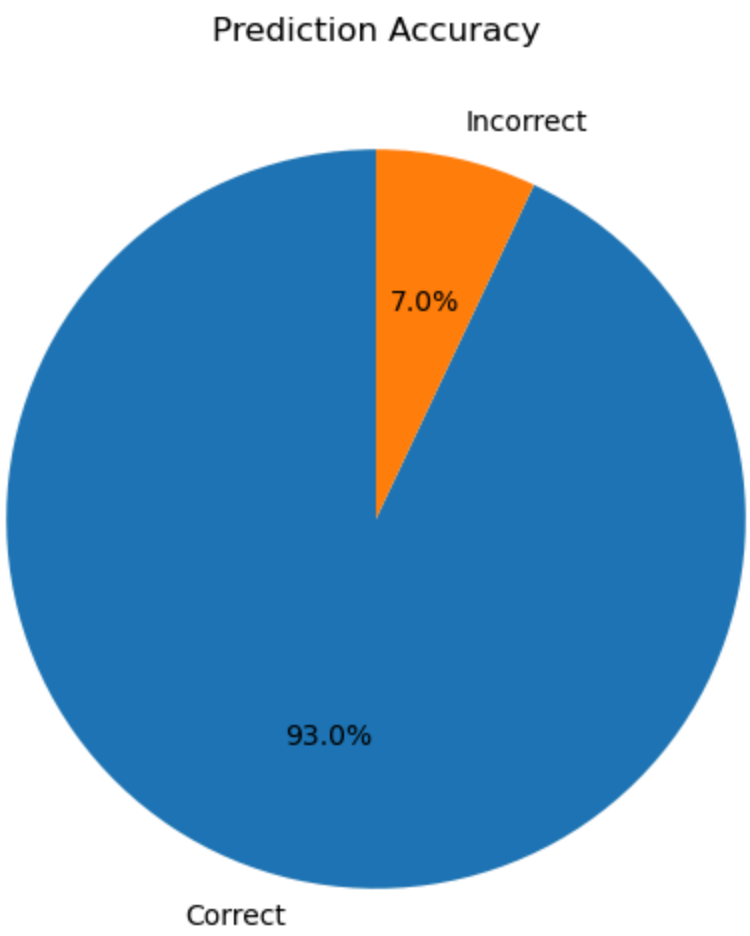
In [33]:
```
labels = ['glioma_tumor','no_tumor','meningioma_tumor','pituitary_tumor']
```

In [35]:
```
predictions = model.predict(X_test)
pred_labels = np.argmax(predictions, axis=1)
true_labels = np.argmax(y_test, axis=1)

plt.figure(figsize=(25, 25))
for i in range(64):
    plt.subplot(8,8, i + 1)
    plt.imshow(X_test[i])
    plt.title(f"V: {labels[true_labels[i]]}\nP: {labels[pred_labels[i]]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

11/11 ━━━━━━━━━━━━━━━━━━━━ **4s** 327ms/step



In [37]:
```
correct_predictions = np.sum(pred_labels == true_labels)
incorrect_predictions = np.sum(pred_labels != true_labels)

plt.figure(figsize=(6, 6))
plt.pie([correct_predictions, incorrect_predictions], labels=['Correct', 'In
plt.title('Prediction Accuracy')
plt.show()
```



In [39]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import warnings
warnings.filterwarnings('ignore')
```

In [41]:
```
classes = {
    'no_tumor': 0,
    'glioma_tumor': 1,
    'meningioma_tumor': 1,
    'pituitary_tumor': 1
}
```

In [43]:
```
import os
import cv2

X = []
Y = []
base_path = r"C:\Users\AISWARYA\Documents\Training"

# Process each class folder
for cls in classes:
```

```
    pth = os.path.join(base_path, cls)  # Correct path
    if os.path.exists(pth):  # Check if directory exists
        for j in os.listdir(pth):  # Loop through files in the directory
            img_path = os.path.join(pth, j)
            img = cv2.imread(img_path, 0)  # Load image in grayscale
            if img is not None:  # Ensure the image is read successfully
                img = cv2.resize(img, (200, 200))  # Resize to 200x200
                X.append(img)
                Y.append(classes[cls])  # Add corresponding label
            else:
                print(f"Failed to read image: {img_path}")
    else:
        print(f"Directory not found: {pth}")
```

In [45]:
```
X = np.array(X)
Y = np.array(Y)
```

In [47]:
```
np.unique(Y)
```

Out[47]:
```
array([0, 1])
```

In [49]:
```
pd.Series(Y).value_counts()
```

Out[49]:
```
1    2475
0     395
Name: count, dtype: int64
```

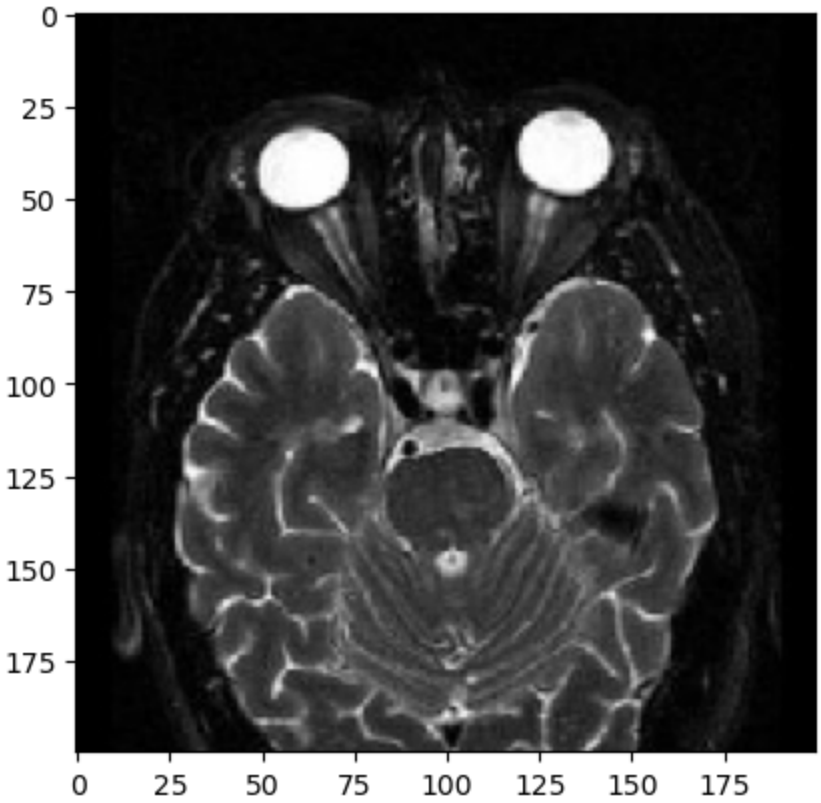
In [51]:
```
X.shape
```

Out[51]:
```
(2870, 200, 200)
```

Visualize data

In [54]:
```
plt.imshow(X[1], cmap='gray')
```

Out[54]:
```
<matplotlib.image.AxesImage at 0x281b9338740>
```



Prepare data

In [57]:
```
X_updated = X.reshape(len(X), -1)
X_updated.shape
```

Out[57]:
```
(2870, 40000)
```

Split Data

In [60]:
```
xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, random_state=1
                                                test_size=.20)
```

In [62]:
```
xtrain.shape, xtest.shape
```

Out[62]:
```
((2296, 40000), (574, 40000))
```

Feature Scaling

In [65]:
```
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
xtrain = xtrain/255
xtest = xtest/255
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
```

```
255 0
255 0
1.0 0.0
1.0 0.0
```

Feature Selection: PCA

In [68]:
```
from sklearn.decomposition import PCA
```

In [70]:
```
print(xtrain.shape, xtest.shape)

pca = PCA(.98)
# pca_train = pca.fit_transform(xtrain)
# pca_test = pca.transform(xtest)
pca_train = xtrain
pca_test = xtest
```

```
(2296, 40000) (574, 40000)
```

Train Model

In [73]:
```
sv = SVC()
sv.fit(pca_train, ytrain)
```

Out[73]:

SVC

SVC()

Evaluation

In [75]:
```
print("Training Score:", sv.score(pca_train, ytrain))
print("Testing Score:", sv.score(pca_test, ytest))
```

```
Training Score: 0.9804006968641115
Testing Score: 0.9442508710801394
```

Prediction

In [77]:
```
pred = sv.predict(pca_test)
np.where(ytest!=pred)
```

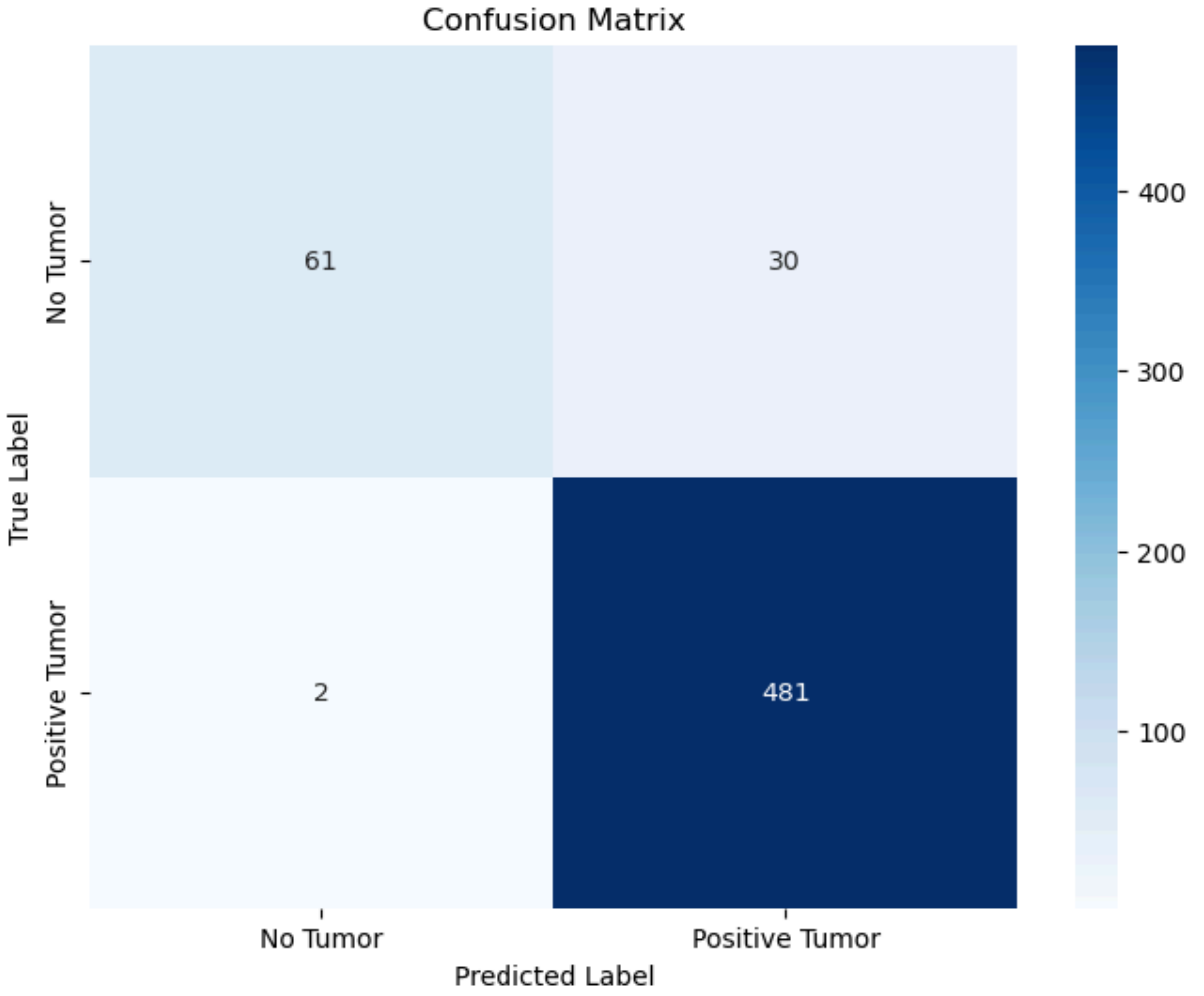
Out[77]:
```
(array([  9,  32,  37,  76, 125, 129, 140, 146, 149, 151, 166, 174, 181,
        187, 203, 242, 313, 338, 356, 375, 423, 438, 441, 448, 453, 466,
        474, 476, 501, 512, 513, 522], dtype=int64),)
```

TEST MODEL

In [79]:
```
dec = {0:'No Tumor', 1:'Positive Tumor'}
```

In [81]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Generate the confusion matrix
```

```
cm = confusion_matrix(ytest, pred)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Tumor',
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [86]:

```
import os
import cv2
import numpy as np
from sklearn.metrics import accuracy_score

# Assuming 'sv' is your trained model and 'dec' is the label decoder
# Define the class labels (decoding)
classes = {'no_tumor': 0, 'pituitary_tumor': 1, 'meningioma_tumor': 1, 'glio
dec = {0:'No Tumor', 1:'Positive Tumor'}
# Create lists to store the actual labels and predictions
y_true = []  # Actual labels
y_pred = []  # Predicted labels

# Path to the test images (you may change this path based on your dataset)
base_path = r"C:\Users\AISWARYA\Documents\Training"
```

```
# Iterate through each class folder
for cls, label in classes.items():
    folder_path = os.path.join(base_path, cls)

    for i in os.listdir(folder_path):
        # Construct the image path
        img_path = os.path.join(folder_path, i)

        # Read the image in grayscale
        img = cv2.imread(img_path, 0)
        if img is not None:
            # Resize the image to 200x200 pixels
            img_resized = cv2.resize(img, (200, 200))

            # Flatten and normalize the image for prediction
            img_flat = img_resized.reshape(1, -1) / 255.0

            # Predict the class using your model (sv)
            pred = sv.predict(img_flat)

            # Append the true and predicted labels
            y_true.append(label)
            y_pred.append(pred[0])

# Calculate the accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 97.32%
```

In [87]:

```
import os
import cv2
import numpy as np
from sklearn.metrics import accuracy_score

# Assuming 'sv' is your trained model and 'dec' is the label decoder
# Define the class labels (decoding)
classes = {'no_tumor': 0, 'pituitary_tumor': 1, 'meningioma_tumor': 1, 'glio
dec = {0:'No Tumor', 1:'Positive Tumor'}

# Create lists to store the actual labels and predictions
y_true = []  # Actual labels
y_pred = []  # Predicted labels

# Path to the test images (you may change this path based on your dataset)
base_path = r""C:\Users\AISWARYA\Documents\Testing"

# Iterate through each class folder
for cls, label in classes.items():
    folder_path = os.path.join(base_path, cls)

    for i in os.listdir(folder_path):
        # Construct the image path
        img_path = os.path.join(folder_path, i)
```

```
        # Read the image in grayscale
        img = cv2.imread(img_path, 0)
        if img is not None:
            # Resize the image to 200x200 pixels
            img_resized = cv2.resize(img, (200, 200))

            # Flatten and normalize the image for prediction
            img_flat = img_resized.reshape(1, -1) / 255.0

            # Predict the class using your model (sv)
            pred = sv.predict(img_flat)

            # Append the true and predicted labels
            y_true.append(label)
            y_pred.append(pred[0])

# Calculate the accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

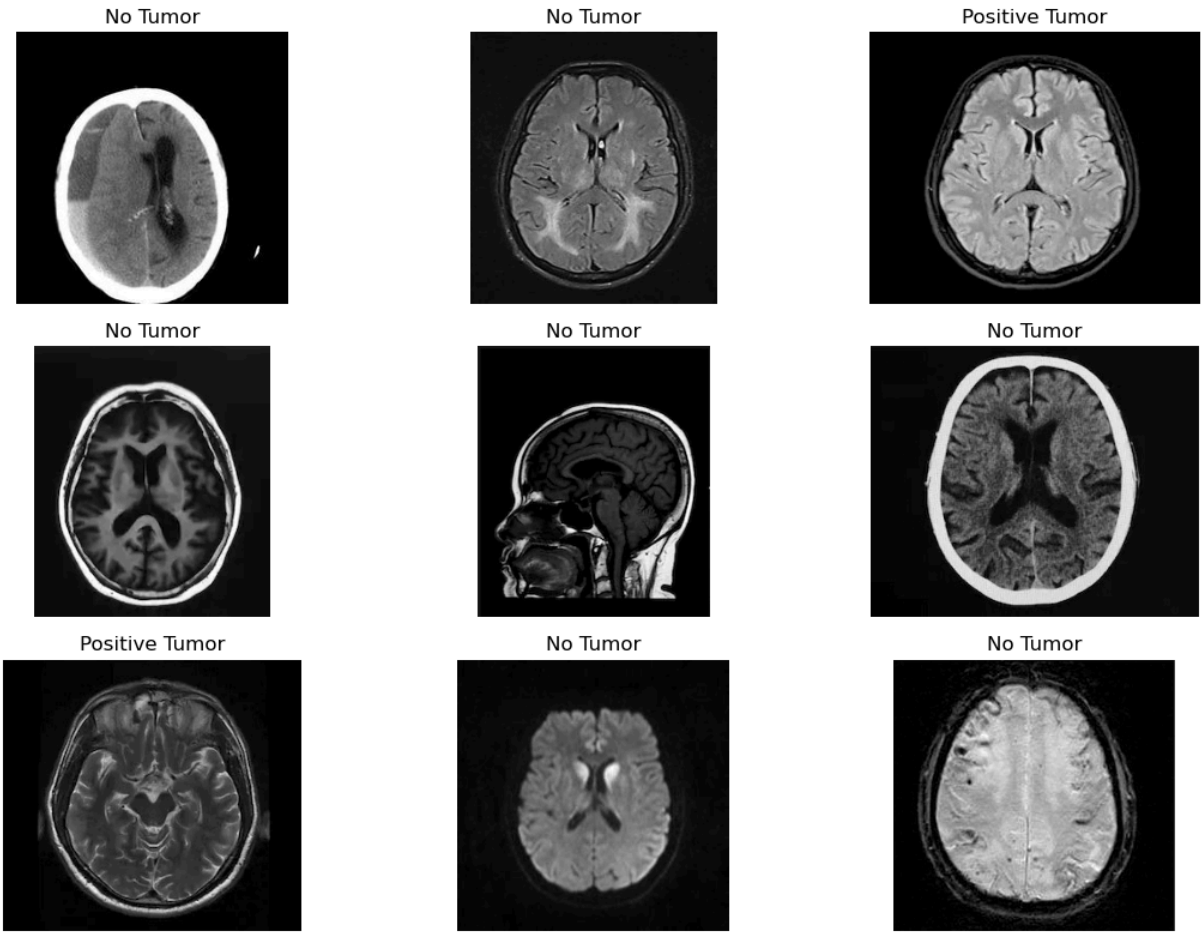
```
Accuracy: 87.31%
```

In [90]:

```
import os
import cv2
import matplotlib.pyplot as plt

# Ensure that the figure displays properly
%matplotlib inline

plt.figure(figsize=(12, 8))
c = 1

# Path to the 'no_tumor' folder
base_path = r"C:\Users\AISWARYA\Documents\Testing\no_tumor"

# Loop through the first 9 images in the folder
for i in os.listdir(base_path)[:9]:
    plt.subplot(3, 3, c)

    # Construct the correct path for each image
    img_path = os.path.join(base_path, i)

    # Load the image in grayscale
    img = cv2.imread(img_path, 0)

    # Check if the image is loaded successfully
    if img is not None:
        print(f"Image {img_path} loaded successfully")  # Debug: confirm ima
        # Resize the image to 200x200 pixels
        img1 = cv2.resize(img, (200, 200))

        # Flatten and normalize the image for prediction
        img1 = img1.reshape(1, -1) / 255.0

        # Predict the class using your model (sv)
        p = sv.predict(img1)
```

```
        # Display the prediction title and the image
        plt.title(dec[p[0]])  # dec is the decoder for labels
        plt.imshow(img, cmap='gray')
        plt.axis('off')
        c += 1
    else:
        print(f"Failed to load image: {img_path}")  # Debug: if the image is

# Ensure the layout is adjusted properly
plt.tight_layout()
plt.show()  # Display the plot
```

```
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(1).jpg loaded succes
sfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(10).jpg loaded succe
ssfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(100).jpg loaded succ
essfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(101).jpg loaded succ
essfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(102).jpg loaded succ
essfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(103).jpg loaded succ
essfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(104).jpg loaded succ
essfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(11).jpg loaded succe
ssfully
Image C:\Users\AISWARYA\Documents\Testing\no_tumor\image(12).jpg loaded succe
ssfully
```

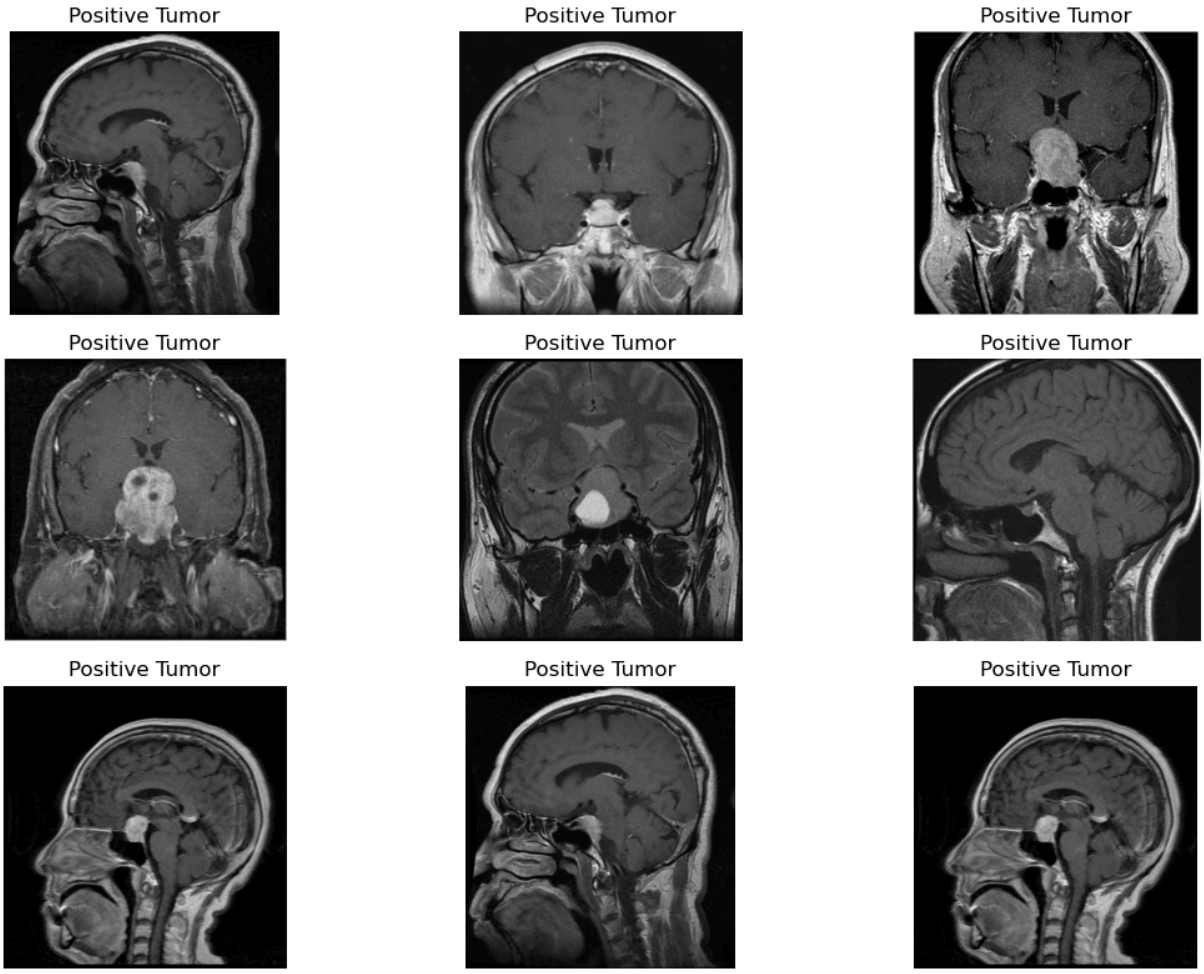


In [92]:

```
import os
import cv2
import matplotlib.pyplot as plt

# Ensure that the figure displays properly
%matplotlib inline

plt.figure(figsize=(12, 8))
c = 1

# Path to the 'no_tumor' folder
base_path = r"C:\Users\AISWARYA\Documents\Testing\pituitary_tumor"

# Loop through the first 9 images in the folder
for i in os.listdir(base_path)[:9]:
    plt.subplot(3, 3, c)

    # Construct the correct path for each image
    img_path = os.path.join(base_path, i)

    # Load the image in grayscale
    img = cv2.imread(img_path, 0)

    # Check if the image is loaded successfully
    if img is not None:
        print(f"Image {img_path} loaded successfully")  # Debug: confirm ima
        # Resize the image to 200x200 pixels
```

```
        img1 = cv2.resize(img, (200, 200))

        # Flatten and normalize the image for prediction
        img1 = img1.reshape(1, -1) / 255.0

        # Predict the class using your model (sv)
        p = sv.predict(img1)

        # Display the prediction title and the image
        plt.title(dec[p[0]])  # dec is the decoder for labels
        plt.imshow(img, cmap='gray')
        plt.axis('off')
        c += 1
    else:
        print(f"Failed to load image: {img_path}")  # Debug: if the image is

# Ensure the layout is adjusted properly
plt.tight_layout()
plt.show()  # Display the plot
```

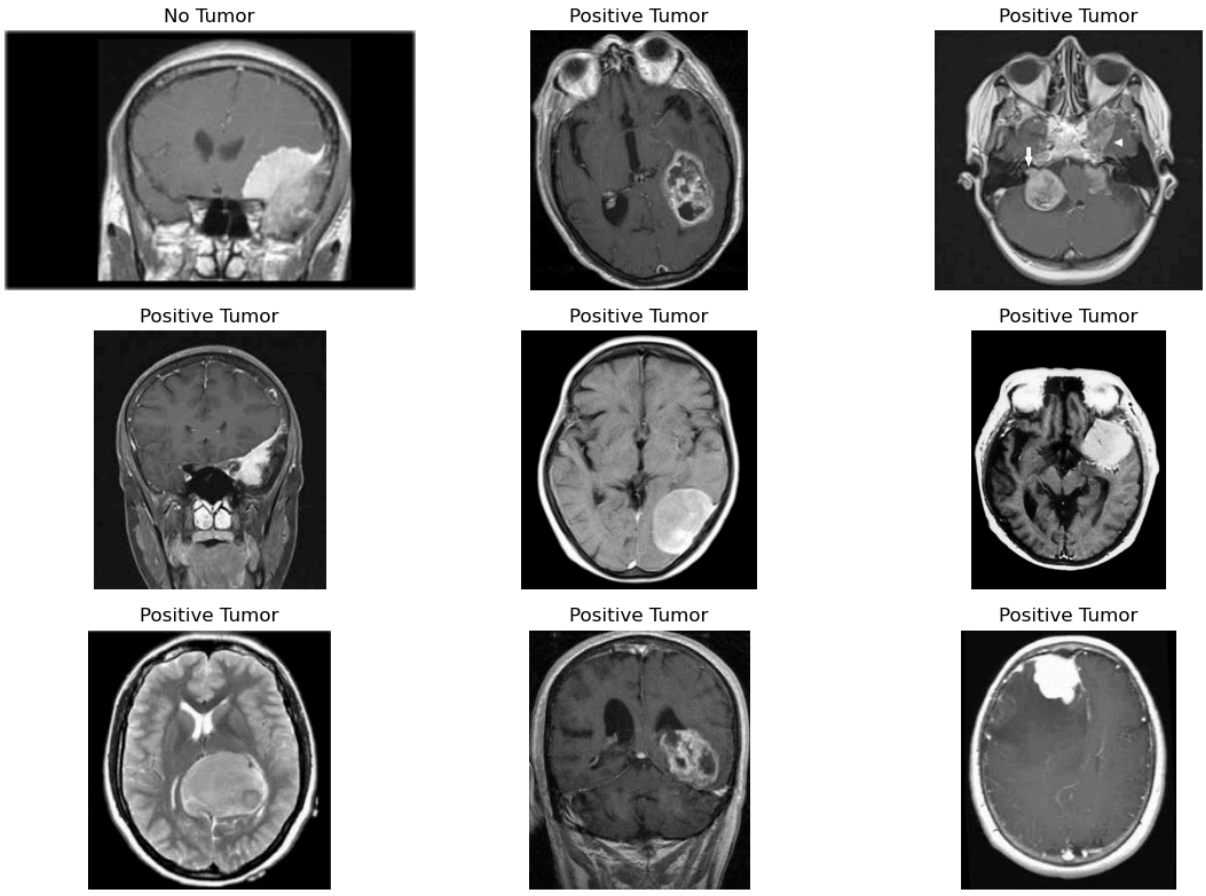
```
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(1).jpg loaded
successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(10).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(11).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(13).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(15).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(18).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(19).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(2).jpg loaded
successfully
Image C:\Users\AISWARYA\Documents\Testing\pituitary_tumor\image(20).jpg loade
d successfully
```



In [94]:
```python
import os
import cv2
import matplotlib.pyplot as plt

# Ensure that the figure displays properly
%matplotlib inline

plt.figure(figsize=(12, 8))
c = 1

# Path to the 'no_tumor' folder
base_path = r"C:\Users\AISWARYA\Documents\Testing\meningioma_tumor"

# Loop through the first 9 images in the folder
for i in os.listdir(base_path)[:9]:
    plt.subplot(3, 3, c)

    # Construct the correct path for each image
    img_path = os.path.join(base_path, i)

    # Load the image in grayscale
    img = cv2.imread(img_path, 0)

    # Check if the image is loaded successfully
    if img is not None:
        print(f"Image {img_path} loaded successfully")  # Debug: confirm ima
```

```
        # Resize the image to 200x200 pixels
        img1 = cv2.resize(img, (200, 200))

        # Flatten and normalize the image for prediction
        img1 = img1.reshape(1, -1) / 255.0

        # Predict the class using your model (sv)
        p = sv.predict(img1)

        # Display the prediction title and the image
        plt.title(dec[p[0]])  # dec is the decoder for labels
        plt.imshow(img, cmap='gray')
        plt.axis('off')
        c += 1
    else:
        print(f"Failed to load image: {img_path}")  # Debug: if the image is

# Ensure the layout is adjusted properly
plt.tight_layout()
plt.show()  # Display the plot
```

```
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(1).jpg loade
d successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(10).jpg load
ed successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(100).jpg loa
ded successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(102).jpg loa
ded successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(106).jpg loa
ded successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(107).jpg loa
ded successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(109).jpg loa
ded successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(11).jpg load
ed successfully
Image C:\Users\AISWARYA\Documents\Testing\meningioma_tumor\image(112).jpg loa
ded successfully
```

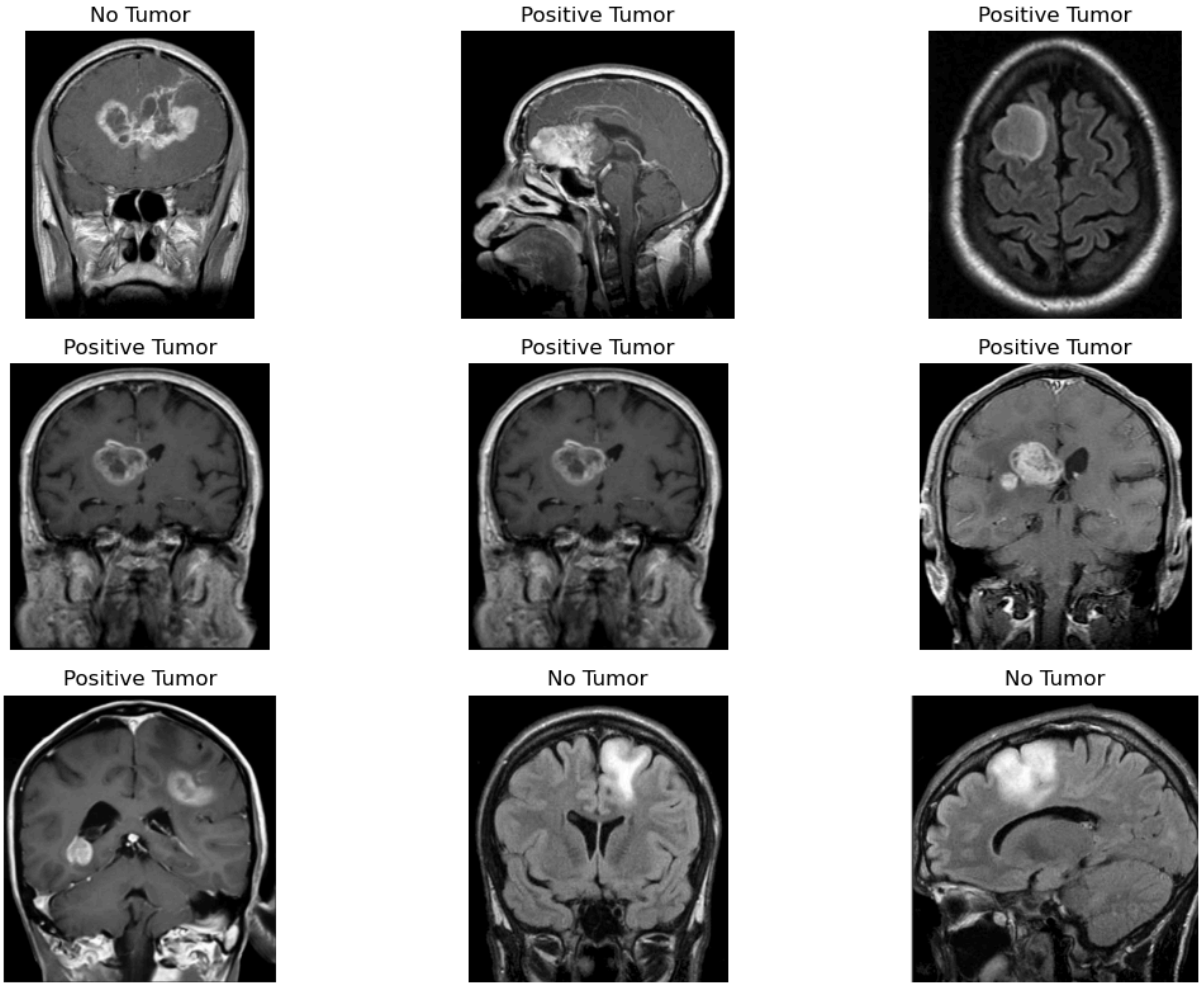


In [96]:

```
import os
import cv2
import matplotlib.pyplot as plt

# Ensure that the figure displays properly
%matplotlib inline

plt.figure(figsize=(12, 8))
c = 1

# Path to the 'no_tumor' folder
base_path = r"C:\Users\AISWARYA\Documents\Testing\glioma_tumor"

# Loop through the first 9 images in the folder
for i in os.listdir(base_path)[:9]:
    plt.subplot(3, 3, c)

    # Construct the correct path for each image
    img_path = os.path.join(base_path, i)

    # Load the image in grayscale
    img = cv2.imread(img_path, 0)

    # Check if the image is loaded successfully
    if img is not None:
        print(f"Image {img_path} loaded successfully")  # Debug: confirm ima
        # Resize the image to 200x200 pixels
        img1 = cv2.resize(img, (200, 200))
```

```
        # Flatten and normalize the image for prediction
        img1 = img1.reshape(1, -1) / 255.0

        # Predict the class using your model (sv)
        p = sv.predict(img1)

        # Display the prediction title and the image
        plt.title(dec[p[0]])  # dec is the decoder for labels
        plt.imshow(img, cmap='gray')
        plt.axis('off')
        c += 1
    else:
        print(f"Failed to load image: {img_path}")  # Debug: if the image is

# Ensure the layout is adjusted properly
plt.tight_layout()
plt.show()  # Display the plot
```

```
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(1).jpg loaded su
ccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(10).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(100).jpg loaded
successfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(11).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(12).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(13).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(14).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(15).jpg loaded s
uccessfully
Image C:\Users\AISWARYA\Documents\Testing\glioma_tumor\image(16).jpg loaded s
uccessfully
```


No Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
No Tumor
No Tumor

In [ ]: