In [11]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing import image  # Import the image functions
import numpy as np
```

In [12]:

```
# Path to dataset
train_dir = 'train'  # Train directory
test_dir = 'test'      # Test directory
```

In [13]:

```
#Define image dimensions
img_width, img_height = 150, 150
```

In [14]:

```
# Image preprocessing
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [15]:

```
# Load images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    shuffle=False  # Important for getting correct labels
)
```

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

In [17]:

```
# Define the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 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(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4, activation='softmax')  # 4 categories
])
```

```
C:\Users\bonde\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. W
hen using Sequential models, prefer using an `Input(shape)` object as the first laye
r in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [18]:

```
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [19]:

```
# Train the model
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator
)
```

```
Epoch 1/10
```

```
C:\Users\bonde\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_datas
et_adapter.py:122: UserWarning: Your `PyDataset` class should call `super().__init__
(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessi
ng`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignor
ed.
  self._warn_if_super_not_called()
```

```
179/179 ━━━━━━━━━━━━━━━━━━━━ 95s 493ms/step - accuracy: 0.5813 - loss: 0.9681 - val_
accuracy: 0.8024 - val_loss: 0.4931
Epoch 2/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 428ms/step - accuracy: 0.8234 - loss: 0.4649 - val_
accuracy: 0.8391 - val_loss: 0.3834
Epoch 3/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 427ms/step - accuracy: 0.8704 - loss: 0.3569 - val_
accuracy: 0.8810 - val_loss: 0.3035
Epoch 4/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 79s 430ms/step - accuracy: 0.9077 - loss: 0.2451 - val_
accuracy: 0.8902 - val_loss: 0.2873
Epoch 5/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 430ms/step - accuracy: 0.9277 - loss: 0.1969 - val_
accuracy: 0.9146 - val_loss: 0.2267
Epoch 6/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 426ms/step - accuracy: 0.9457 - loss: 0.1444 - val_
accuracy: 0.9100 - val_loss: 0.2042
Epoch 7/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 79s 432ms/step - accuracy: 0.9553 - loss: 0.1248 - val_
accuracy: 0.9451 - val_loss: 0.1650
Epoch 8/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 426ms/step - accuracy: 0.9605 - loss: 0.1136 - val_
accuracy: 0.9489 - val_loss: 0.1504
Epoch 9/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 427ms/step - accuracy: 0.9628 - loss: 0.0940 - val_
accuracy: 0.9497 - val_loss: 0.1697
Epoch 10/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 78s 429ms/step - accuracy: 0.9646 - loss: 0.0941 - val_
accuracy: 0.9527 - val_loss: 0.1552
```

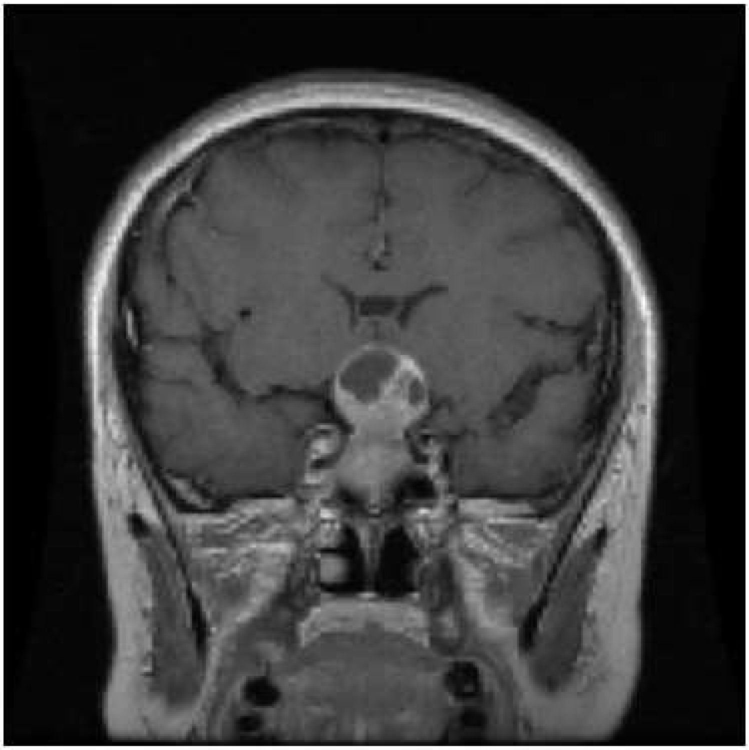
In [20]:

```python
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test accuracy: {test_acc}")

# Make predictions on the test set
predictions = model.predict(test_generator)
predicted_classes = tf.argmax(predictions, axis=1)  # Get the predicted class indic
print(predicted_classes)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 9s 214ms/step - accuracy: 0.9240 - loss: 0.2431
Test accuracy: 0.952707827091217
41/41 ━━━━━━━━━━━━━━━━━━━━ 9s 215ms/step
tf.Tensor([0 0 0 ... 3 3 3], shape=(1311,), dtype=int64)
```

In [27]:

```python
def preprocess_image(img_path, img_width, img_height):
    img = image.load_img(img_path, target_size=(img_width, img_height))  # Load and
    img_array = image.img_to_array(img)  # Convert image to array
    img_array = np.expand_dims(img_array, axis=0)  # Add a batch dimension
    img_array /= 255.0  # Normalize the image
    return img_array

img_path = r'C:\Users\bonde\Desktop\dl_mock\7\train\pituitary\Tr-pi_0011.jpg'
preprocessed_img = preprocess_image(img_path, img_width, img_height)

plt.imshow(image.load_img(img_path, target_size=(img_width, img_height)))
plt.axis('off')  # Turn off axis
plt.show()

# Make predictions
prediction = model.predict(preprocessed_img)

# Get the predicted class index
predicted_class = np.argmax(prediction, axis=1)

# Class labels
class_labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor','pituitary_tumor']

# Get the class label for the predicted class
predicted_label = class_labels[predicted_class[0]]

# Output the prediction
print(f'The model predicts: {predicted_label}')
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
The model predicts: pituitary_tumor
```

In [ ]: