```
from keras.utils import to_categorical
import cv2
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from keras.models import load_model
from skimage import feature


def calculate_lbp(image, num_points=8, radius=3):
    try:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        lbp = feature.local_binary_pattern(gray, num_points, radius, method="uniform")
        hist, _ = np.histogram(lbp.ravel(), bins=num_points + 2, range=(0, num_points + 2))
        hist = hist.astype("float")
        hist /= (hist.sum() + 1e-7)
        return hist
    except Exception as e:
        print(f"Error in calculate_lbp: {e}")
        return None


# from google.colab import drive
# drive.mount('/content/drive')


# Load the pre-trained model
model = load_model('/content/drive/MyDrive/test/Creation_model.h5')


# Path to the directory containing test images
test_data_dir = "/content/drive/MyDrive/test/train"


# Get a random category
categories = ["Closed", "Open", "no_yawn", "yawn"]
random_category = random.choice(categories)


# Get a random image from the random category
random_image_path = os.path.join(test_data_dir, random_category, random.choice(os.listdir(os.path.join(test_data_dir, random_category))))


img = cv2.imread(random_image_path)
if img is None:
    print(f"Failed to read image at path: {random_image_path}")
else:
    # Resize the image if needed
    your_target_width = 224
    your_target_height = 224
    img = cv2.resize(img, (your_target_width, your_target_height))

    # Calculate LBP features
    features = calculate_lbp(img)

    # Pad the features to have the same length
    max_hist_length = model.input_shape[1]
    features_padded = np.pad(features, (0, max_hist_length - len(features)), constant_values=0)

    # Convert to numpy array and reshape for CNN input
    X_test = np.array(features_padded).reshape(1, max_hist_length, 1)

    # Assuming you have a dataset with features X_test and labels y_test
    # Replace this line with your actual code to load test labels
    # Predict the category
    prediction = model.predict(X_test)
    predicted_category = np.argmax(prediction)

    # Map numerical index to category label
    category_labels = ["Closed", "Open", "no_yawn", "yawn"]
    predicted_category_label = category_labels[predicted_category]

    # Get the filename from the path
    file_name = os.path.basename(random_image_path)

    # Visualize the image and predicted category in the file name
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
plt.title(f"File: {file_name}\nTrue Category: {random_category}\nPredicted Category: {predicted_category_label}")
plt.show()
```

```
1/1 [==============================] - 0s 19ms/step
```