Test Plan for Image Comparison Script

Submitted by: Walid Adel

GitHub Repository Link

Table of Contents

Introduction

This test plan covers all **25 test cases** from the image comparison script to ensure its functionality, error handling, and performance. It includes functional tests for different scenarios such as identical images, tolerance-based differences, error handling, and output generation.

Test Environment

The testing was carried out on a machine with the following specifications:

- **Operating System**: Ubuntu 20.04
- **Python Version**: 3.8
- **Hardware**: 16GB RAM, 4-core CPU, NVIDIA GPU for large image processing tests
- **Libraries**:
  - Pillow
  - NumPy
  - scikit-image
  - pytest
  - pytest-html
  - pytest-cov

  Install dependencies:

  ```
  pip install pillow numpy scikit-image pytest pytest-html pytest-cov
  ```

Test Cases

Test Group for color_similarity_detection_technique.py

Test Case 1: Identical Images

- **Test Data**: `image1.jpg` and `image1_copy.jpg`
- **Steps**: Compare identical images with 0% tolerance.
- **Expected Output**: No differences, 0 differing pixels in the report.
- **Covered Requirements**: Identical image comparison.

Test Case 2: Small Differences

- **Test Data**: `image1.jpg` and `image2_small_diff.jpg`
- **Steps**: Compare images with minor pixel differences, 0% tolerance.
- **Expected Output**: Small pixel differences detected, differences reported.
- **Covered Requirements**: Detecting small changes.

Test Case 3: Tolerance-Based Comparison

- **Test Data**: `image1.png` and `image2_tolerance.png`

- **Steps**: Run with 0%, 5%, and 10% tolerance.
- **Expected Output**: Differences detected according to tolerance levels.
- **Covered Requirements**: Tolerance-based comparison.

Test Case 4: Completely Different Images

- **Test Data**: image1.jpg and image3_different.jpg
- **Steps**: Compare two completely different images.
- **Expected Output**: All pixels are marked as different.
- **Covered Requirements**: Handling completely different images.

Test Case 5: Invalid Image Formats

- **Test Data**: image1.jpg and invalid_image.pdf
- **Steps**: Compare a valid image with an invalid format.
- **Expected Output**: Error raised for invalid format.
- **Covered Requirements**: Invalid format error handling.

Test Case 6: Different Image Sizes

- **Test Data**: image1.jpg and image4_different_size.jpg
- **Steps**: Compare images of different sizes.
- **Expected Output**: Error raised for different sizes.
- **Covered Requirements**: Size mismatch error handling.

Test Case 7: Corrupted Image File

- **Test Data**: corrupted_image1.jpg and image2.jpg
- **Steps**: Compare a corrupted image with a valid image.
- **Expected Output**: Error raised for corrupted image.
- **Covered Requirements**: Handling corrupted files.

Test Case 8: Missing Image File

- **Test Data**: non_existing_image.jpg and image1.jpg
- **Steps**: Run with a non-existing file.
- **Expected Output**: Error raised for missing file.
- **Covered Requirements**: Missing file handling.

Test Case 9: Correct Report Generation

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Generate a comparison report.
- **Expected Output**: Correct report with pixel differences.
- **Covered Requirements**: Report generation.

Test Case 10: Image Output for RGB

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Generate difference images for RGB.
- **Expected Output**: RGB difference images are saved.
- **Covered Requirements**: RGB output generation.

Test Case 11: Image Output for Grayscale

- **Test Data**: image1_grayscale.jpg and image2_grayscale.jpg
- **Steps**: Compare grayscale images.
- **Expected Output**: Grayscale difference images are saved.
- **Covered Requirements**: Grayscale output generation.

Test Case 12: Grayscale vs RGB Comparison

- **Test Data**: image1_grayscale.jpg and image1.jpg
- **Steps**: Compare grayscale and RGB images.
- **Expected Output**: Error raised for mismatched formats.
- **Covered Requirements**: Image format mismatch handling.

Test Case 13: Similar Grayscale Images

- **Test Data**: image1_grayscale.jpg and image1_grayscale.jpg
- **Steps**: Compare similar grayscale images.
- **Expected Output**: No differences detected.
- **Covered Requirements**: Grayscale image comparison.

Test Case 14: Different Grayscale Images

- **Test Data**: image1_grayscale.jpg and image2_grayscale.jpg
- **Steps**: Compare different grayscale images.
- **Expected Output**: Differences detected.
- **Covered Requirements**: Grayscale image difference detection.

Test Case 15: Small 200x200 Images

- **Test Data**: image1_200x200.jpg and image2_200x200.jpg
- **Steps**: Compare small images (200x200 pixels).
- **Expected Output**: Differences detected, performance analyzed.
- **Covered Requirements**: Small image comparison.

Test Case 16: Large 4000x4000 Images

- **Test Data**: image1_4000x4000.jpg and image2_4000x4000.jpg
- **Steps**: Compare large images (4000x4000 pixels).
- **Expected Output**: Differences detected, performance analyzed.
- **Covered Requirements**: Large image comparison.

Test Group for image_compare.py

Test Case 17: Valid Image Comparison

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Run via the main function.
- **Expected Output**: Comparison successfully performed.
- **Covered Requirements**: Main function handling.

Test Case 18: Invalid Image Format

- **Test Data**: image1.jpg and invalid_image.pdf
- **Steps**: Run main function with an invalid image format.
- **Expected Output**: Error raised for invalid format.
- **Covered Requirements**: Invalid format error handling in main function.

Test Case 19: Missing Image File

- **Test Data**: non_existing_image.jpg and image1.jpg
- **Steps**: Run main function with missing image.
- **Expected Output**: Error raised for missing file.
- **Covered Requirements**: Missing file handling in main function.

Test Case 20: Different Image Formats

- **Test Data**: image1.jpg and image1.png
- **Steps**: Run main function with different image formats.
- **Expected Output**: Error raised for mismatched formats.
- **Covered Requirements**: Format mismatch handling in main function.

Test Case 21: Valid Tolerance Value

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Run with a valid tolerance value.
- **Expected Output**: Successful comparison with tolerance.
- **Covered Requirements**: Valid tolerance handling.

Test Case 22: Invalid Tolerance Value

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Run with an invalid tolerance value (e.g., 120%).
- **Expected Output**: Error raised for invalid tolerance.
- **Covered Requirements**: Invalid tolerance handling.

Test Case 23: Invalid Image File Opening

- **Test Data**: corrupted_image1.jpg

- **Steps**: Attempt to open a corrupted image file.
- **Expected Output**: Error raised for corrupted image.
- **Covered Requirements**: Invalid image handling.

Test Case 24: Non-Numeric Tolerance Value

- **Test Data**: image1.jpg and image1.jpg
- **Steps**: Run with a non-numeric tolerance value.
- **Expected Output**: Error raised for non-numeric tolerance.
- **Covered Requirements**: Non-numeric tolerance handling.

Test Case 25: Argument Handling via sys.argv

- **Test Data**: image1.jpg and image2.jpg
- **Steps**: Simulate running with arguments via sys.argv.
- **Expected Output**: Arguments are parsed and handled correctly.
- **Covered Requirements**: Argument parsing.

Running the Tests

1. **Generate Test Data**:
    - Use the test_data_generator.py script to generate necessary test images:

```
python test_data_generator.py
```

2. **Run Automated Tests**:
    - Use pytest to execute the test cases:

```
pytest automated_test_cases.py
```

3. **Generate HTML Report**:
    - Generate an HTML report for test results:

```
pytest --html=../output/test_report.html --self-contained-html automated_test_cases.py
```

4. **Measure Code Coverage**:

```
coverage run --source=../src -m pytest automated_test_cases.py
coverage html -d ../output/coverage
```

Expected Output and Reports

- **Difference Images**:

    - output/diff_img1.png: Pixels different only in the first image.
    - output/diff_img2.png: Pixels different only in the second image.
    - output/combined_diff.png: Combined differences from both images.
- **Statistical Report**:
    - output/comparison_report.txt: Includes total pixels, differing pixels, and percentage of differing pixels.
- **Test Report**:
    - output/test_report.html: Summary of all test case results.
- **Code Coverage**:
    - output/coverage/index.html: Coverage analysis for the source files.