**Unit Testing**

Test Case ID: TFM\_UT\_FT\_1

Test Case Title: Compare Two Identical Text Files

Test Category: Unit Testing - Functional Testing

Author: Wing Cheung Chow

Created Date: 1 February 2025

Last Updated: 1 February 2025

Priority: High

Status: Approved

Test Case Description: To verify that the file comparison functionality correctly identifies that two text files with same content should have no differences.

Pre-Conditions:

Two test files with the same content exist.

Test Steps:

Initialize the file path for test data.

Compare the expected result with output from CompareFiles function.

Observe the test results in the test environment.

Test Data:

FT\_IdenticalFile1.txt and FT\_IdenticalFile2.txt.

Expected Results:

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As expected.

Test Status: Passed

Test Case ID: TFM\_UT\_FT\_2

Test Case Title: Compare Two Different Text Files

Test Category: Unit Testing - Functional Testing

Author: Wing Cheung Chow

Created Date: 1 February 2025

Last Updated: 1 February 2025

Priority: High

Status: Approved

Test Case Description: To verify that the file comparison functionality correctly identifies that two text files with different content should have differences.

Pre-Conditions:

Two test files with different content exist.

Test Steps:

Initialize the file path for test data.

Compare the expected result with output from CompareFiles function.

Observe the test results in the test environment.

Test Data:

FT\_DiffFile1.txt and FT\_DiffFile2.txt.

Expected Results:

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As expected.

Test Status: Passed

Test Case ID: TFM\_UT\_BT\_1

Test Case Title: Compare an Empty File with a Non-Empty File

Test Category: Unit Testing - Boundary Testing

Author: Wing Cheung Chow

Created Date: 1 February 2025

Last Updated: 2 February 2025

Priority: High

Status: Draft

Test Case Description: To verify that when comparing an empty file with a non-empty file, all lines from the non-empty file are correctly detected as new/added, and the application does not crash or exhibit unexpected behavior.

Pre-Conditions:

Two test files exist: Empty file (0 bytes). Another file contains multiple lines of text.

Test Steps:

Initialize the file path for test data.

Compare the expected result with output from CompareFiles function.

Observe the test results in the test environment.

Test Data:

BT\_EmptyFile.txt and BT\_File2.txt.

Expected Results:

All lines in BT\_File2.txt should be detected as new/added.

No application crashes, errors, or unexpected behavior should occur.

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As Expected.

Test Status: Pass/Fail

**System Testing**

Test Case ID: TFM\_ST\_FT\_1

Test Case Title: Verify that clicking the Compare button initiates file comparison, detects differences and supports .odt and .docx file formats

Test Category: System Testing - Functional Testing

Author: Wing Cheung Chow

Created Date: 1 February 2025

Last Updated: 1 February 2025

Priority: High

Status: Approved

Test Case Description: To ensure that clicking the Compare Files button triggers the file comparison operation and correctly identifies differences between the selected files, and support both .odt and .docx file formats.

Pre-Conditions:

Two test files exist: FT\_DiffFile1.odt (file with content), FT\_DiffFile2.docx (file with different content).

Test Steps:

Initialize the file path for test data.

Simulate clicking the Compare Files button by invoking the button's RaiseEvent method.

Observe the application's response and verify if differences are detected and populated in the UI.

Test Data:

FT\_DiffFile1.odt and FT\_DiffFile2.docx.

Expected Results:

The file comparison operation is triggered successfully.

The application correctly identifies differences between the two files.

The Differences Collection in the UI is populated with detected changes.

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As expected.

Test Status: Passed

Test Case ID: TFM\_ST\_FT\_2

Test Case Title: Verify that the Save Output button successfully saves the comparison result

Test Category: System Testing - Functional Testing

Author: Wing Cheung Chow

Created Date: 1 February 2025

Last Updated: 1 February 2025

Priority: High

Status: Approved

Test Case Description: To ensure that clicking the Save Output button saves the comparison results into a specified output file.

Pre-Conditions:

File comparison has already been performed before clicking Save Output.

Two test files exist: FT\_DiffFile1.txt (file with content), FT\_DiffFile2.txt (file with different content).

Test Steps:

Initialize the file path for test data.

Simulate clicking the Compare Files button by invoking the button's RaiseEvent method.

Set outputFile as the Output File Name programmatically.

Simulate clicking the Save Output button using RaiseEvent.

Observe the application's response and verify if output file saved message is shown in the UI.

Test Data:

FT\_DiffFile1.txt and FT\_DiffFile2.txt.

Expected Results:

The Differences Collection in the UI is populated with detected changes.

The application successfully saves the comparison result in outputFile.txt.

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As expected.

Test Status: Passed

Test Case ID: TFM\_ST\_NT\_1

Test Case Title: Verify that Unsupported File Formats are Handled Properly

Test Category: System Testing - Negative Testing

Author: Wing Cheung Chow

Created Date: 2 February 2025

Last Updated: 2 February 2025

Priority: High

Status: Draft

Test Case Description: To ensure that the application handles unsupported file formats (e.g., .exe, .mp3, .png) correctly by displaying an error message without causing any crashes or unexpected behavior.

Pre-Conditions:

Two unsupported file types exist: NT\_invalid1.exe (Executable file) and NT\_invalid2.png (Image file).

Test Steps:

Initialize the file path for test data.

Simulate clicking the Compare Files button by invoking the button's RaiseEvent method.

Observe the application's response and verify if error message is shown in the UI.

Test Data:

NT\_invalid1.exe and NT\_invalid2.png.

Expected Results:

The application should display an error message indicating that the file formats are not valid.

No comparison should occur.

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

As expected.

Test Status: Pass/Fail

Test Case ID: TFM\_ST\_PT\_1

Test Case Title: Verify Performance of the Application with Large Files

Test Category: System Testing - Performance Testing

Author: Wing Cheung Chow

Created Date: 2 February 2025

Last Updated: 2 February 2025

Priority: High

Status: Draft

Test Case Description: To ensure that the application performs efficiently and without significant delays when comparing large files.

Pre-Conditions:

Two large test files are prepared for comparison: PT\_LargeFile1.txt (e.g., 100MB) and PT\_LargeFile2.txt (e.g., 100MB).

Test Steps:

Initialize the file path for test data.

Simulate clicking the Compare Files button by invoking the button's RaiseEvent method.

Measure the time taken to complete the comparison.

Observe the application's responsiveness during the comparison process.

Verify if comparison completed message is shown in the UI.

Test Data:

PT\_LargeFile1.txt (100MB text file) and PT\_LargeFile2.txt (100MB text file).

Expected Results:

The comparison process should complete within the acceptable time limit

(30 seconds per file).

The application should handle large files without crashes, delays, or errors.

The application should remain responsive and not freeze during the comparison process.

The test results in test explorer should be passed. No errors or crashes occur.

Actual Results:

Time taken: ?? seconds

As expected.

Test Status: Pass/Fail

Test Case ID: TFM\_ST\_UT\_1

Test Case Title: Verify User Navigation and Understanding of Application Features

Test Category: System Testing - Usability Testing

Author: Wing Cheung Chow

Created Date: 2 February 2025

Last Updated: 2 February 2025

Priority: High

Status: Draft

Test Case Description: To ensure that the user can easily navigate the application without confusion and understand how to use its core features, such as file comparison and saving output, without the need for external help or documentation.

Pre-Conditions:

The user has basic knowledge of using software applications.

The main window of the application is displayed with visible buttons like "Browse", "Compare Files", "Save Output" and "Modify File".

The user has no prior experience with the application.

Test Steps:

Launch the Application.

Locate and click the "Compare Files" button in the UI.

Enter valid file paths using "Browse" button and click "Compare Files" button to observe the application's behavior.

Ensure the "Save Output" and "Modify File" buttons are visible and enabled after a comparison is made.

Click between sections and buttons to ensure smooth and intuitive navigation.

Test Data:

UT\_TestFile1.txt and UT\_TestFile2.txt.

Expected Results:

Navigation:

The user should be able to locate and navigate to the "Browse", "Compare Files", "Save Output" and "Modify File" buttons easily.

The navigation between UI elements should be smooth and intuitive, without requiring any instructions or external help.

Ease of Use:

The user should be able to understand how to perform the file comparison feature by interacting with the application’s interface alone.

The process of selecting files, performing a comparison, and saving the output should be straightforward and easily accomplished.

The "Save Output" and "Modify File" buttons should only be enabled once a valid comparison has been completed.

The user should be able to understand how to use the application and finish comparing input files, saving the output file within 10 minutes.

Actual Results:

As expected.

Test Status: Pass/Fail