**1. Basic Functionality Test:**

* **Test**: Use the UI to replace a simple word in all files.
* **Steps**:
  + Use the UI to replace "text" with "word" in all the provided files.
  + Verify that in each file, the word "text" has been replaced by "word":
    - **file1.txt**: *"This is a simple test file. Let's replace some word. Here's more word to test."*
    - **file2.txt**: *"Another file with some word to replace. Let's see how it works. More lines of word here."*
    - **file3.txt**: *"This file doesn't have much word, but it still needs replacement. Just a single match."*

**2. Case Sensitivity Test:**

* **Test**: Test case-insensitive replacements.
* **Steps**:
  + Modify **file1.txt** and add mixed-case occurrences of the word "Text":  
    *"This is a simple test file. Let's replace some Text. Here's more text to test."*
  + Use the UI to replace "text" with "word".
  + Verify that all variations of "text", regardless of case, are replaced:
    - *"This is a simple test file. Let's replace some word. Here's more word to word."*

**3. Log File Creation Test:**

* **Test**: Verify the log file is created and records the replacements.
* **Steps**:
  + Specify a valid log file path (e.g., log.txt) in the UI.
  + Perform the text replacement for "text" to "word".
  + Check the contents of log.txt to confirm that it logs the replacements for **file1.txt**, **file2.txt**, and **file3.txt**.

**4. No Log File Path Specified:**

* **Test**: Ensure the program works correctly without specifying a log file.
* **Steps**:
  + Leave the "Log File Path" field empty.
  + Replace "text" with "word".
  + Verify that the program runs without errors and no log file is generated.

**5. Invalid Log File Path:**

* **Test**: Test how the program behaves with an invalid log file path.
* **Steps**:
  + Enter an invalid path in the "Optional Log File" field (e.g., C:\Invalid\Path\log.txt).
  + Replace "text" with "word".
  + Verify that the program shows an appropriate error message and handles the invalid path gracefully.

**6. Multiple File Test:**

* **Test**: Ensure the program processes multiple files in the directory.
* **Steps**:
  + Place **file1.txt**, **file2.txt**, and **file3.txt** in a directory.
  + Replace "text" with "word".
  + Verify that all files were processed and the word "text" was replaced in all of them.

**7. Empty Fields Test:**

* **Test**: Ensure the program handles empty input fields correctly.
* **Steps**:
  + Leave the "Original Text" field empty in the UI.
  + Try running the program.
  + Verify that the program shows an appropriate error message and doesn't proceed with the replacement.
  + Repeat two more times, 2nd time leaving “New Text” empty, and 3rd time leaving both fields empty.

**8. Special Characters Test:**

* **Test**: Test the program's handling of special characters.
* **Steps**:
  + Add special characters to **file1.txt**:  
    *"Hello, this is a test! Let's replace some #Text."*
  + Use the UI to replace "#Text" with "Groovy".
  + Verify that the special character "#Text" is replaced correctly.

**9. Non-Existing Directory:**

* **Test**: Ensure the program handles non-existing directories gracefully.
* **Steps**:
  + Provide a non-existing directory in the "Directory Path" field (e.g., C:\Nonexistent\Path\).
  + Try running the program.
  + Verify that the program displays an appropriate error message.

**10. Long Text Replacement:**

* **Test**: Replace long phrases in the files.
* **Steps**:
  + Add a longer phrase to **file2.txt**:  
    *"This is a test file with a longer phrase that needs replacing."*
  + Use the UI to replace "longer phrase that needs replacing" with "short text".
  + Verify that the long phrase is replaced correctly.