Bug Tracker for File – Explorer Application

A **bug tracker** is a tool or system used to manage and track issues, bugs, and enhancements during software development. It helps teams keep track of known issues, prioritize fixes, and ensure that nothing is overlooked.

**1. File Operations Logging**:

* The logOperation function currently logs operations to a file and outputs them to the console. If the log file cannot be opened, it only prints an error to the console. You might want to implement a fallback mechanism or alert the user more clearly.
* Consider rotating logs when they reach a certain size to avoid excessive log files.

**Issue: File Operations Logging**

The logOperation function logs operations to a file and outputs them to the console. If the log file cannot be opened, the function only prints an error to the console. There is also a need to implement log rotation to prevent log files from growing excessively large.

**Steps to Reproduce the Issue**

1. **Setup the Environment:**
   * Create or use an existing C++ project with the logOperation function implemented.
   * Ensure the function logs messages to a specified file and outputs to the console.
2. **Reproduce the Log File Opening Failure:**
   * Make the log file path invalid or set file permissions such that the application cannot open or write to the log file.
   * Run the application and invoke logOperation to trigger logging.

**Expected Result: Log File Opening Failure:** If the log file cannot be opened, the function should:Attempt to use a fallback log file or directory. Alert the user with a clear error message, either in the console or via a dialog (depending on the application's interface).

**Actual Result (Without Improvements): Log File Opening Failure:** The application prints an error to the console stating that the log file could be opened.

**Status:** Open

**2. Error Handling:**

In functions like moveFile, copyFile, and deleteFile, error handling could be improved by providing more detailed error messages or even retry mechanisms for certain errors. It might be useful to log errors as well, so they are recorded for troubleshooting.

#### ****Steps to Reproduce****

1. **Move Operation Failure:**
   * Set up a scenario where the source file is open or locked by another process.
   * Try to move the file using moveFile.
2. **Copy Operation Failure:**
   * Set up a scenario where the destination directory is read-only.
   * Try to copy the file using copyFile.
3. **Delete Operation Failure:**
   * Set up a scenario where the file to be deleted is protected by permissions.
   * Try to delete the file using deleteFile.

#### ****3. Expected Result****

* **Detailed Error Messages:** The application should display a detailed error message in the console, specifying the nature of the failure.
* **Error Logging:** The error message should be logged in error\_log.txt for troubleshooting.
* **Optional Retry Mechanism:** If implemented, the function should retry the operation based on specific errors (e.g., temporary file lock).

#### ****4. Actual Result (Before Improvements)****

* The application might only display a generic error message or fail silently.
* No detailed information about the error is logged, making troubleshooting difficult.

#### ****5. Actual Result (After Improvements)****

* **Move Operation:** If the file is locked, the console displays a specific message like "Error moving file: file is locked by another process." The error is logged in error\_log.txt.
* **Copy Operation:** If the destination is read-only, the console displays "Error copying file: permission denied," and the error is logged.
* **Delete Operation:** If the file cannot be deleted due to permissions, the console displays "Error deleting file: permission denied," and the error is logged.

**Status:** Open

### ****Conclusion****

By enhancing error handling in your file operations, you can provide better feedback to users, improve the application's robustness, and make troubleshooting easier with detailed logs. Implementing these changes will also help you identify and resolve issues more efficiently during development and in production.