**Project 4 Mock Repository – ReadMe**

To The User:

- Please be sure to set multiple startup projects, CommServer, CommUser, and GUI and run with VisualStudio.

Requirements and Bonus Items:

- Support browsing /navigating files (Test Drivers and Tested Code Elements) remote Mock Repository

In the GUI Navi/Swap page the user can browse files and navigate directories by clicking on ListBoxItem elements containing names of files and directories in the Mock Repository and the User’s Machine Directory.

- Support selecting / specifying DLL files in the GUI so to run tests on these

In the Build And Test page the user can navigate to files in subdirectories of the Mock Repository. Any file which is a DLL can be added to a list of tested code element files to test, unless they are test driver DLLs, which can be added one at a time to a test request under construction (I.e., one test driver per test request). Any number of tested code elements can be added. Use can clear the test driver elements, or the tested code elements, or both, and add elements at any time before or after submitting the test request.

- Support searching for specified DLL directories or DLL files (by name) which are in the Mock Repository.

In the Navi/Swap page the user can enter the name of a file or directory and choose which to search for. If the named file or directory exists as such in the Mock Repository, it’s path will be displayed in the input field, if not, “File/Directory not found” will be displayed.

- Support building TestRequests using the GUI, by selecting library files from the repository.

As described above, TestRequests are specified and submitted in the Build And Test page of the GUI, then packaged in messages and routed through the user process to the server to be executed.

- Support uploading DLL files

In the Navi/Swap page, the user can navigate to and double-click on items in the User Machine Directory or its subdirectories to populate an upload file name text field and click an “Upload” button to upload files to the current subdirectory of the Mock Repo, relative to that displayed in the GUI. File uploading is implemented in the User process, as their User Machine Directories are only supposed to be accessible from their machine.

- Support creating remote DLL directories (A Mock Repository)

In the Build and Test page the user can enter into a text field a subdirectory name with path relative to the Mock Repo root directory and click a “MakeDir” button to create the directory. Navigation to this directory can take place in the Build And Test page or in the Navi/Swap page.

- Support browsing /navigating/displaying test logs which are in the Mock Repository.

In the Build And Test page and the Navi/Swap page the user can browse all the test logs, these are maintained with unique file names “TestLog\_(test ID)” between runs of the server and/or GUI/Client programs. To display the contents of an individual test log file the user will need to open the file from their native machine file explorer.

- Support searching for specified test log directories or test log files (by name) which are in the Mock Repository.

Searching for specific test log files by name or location (directory) is the same as searching for specified DLL directories or DLL files (by name), which is described above.

- Support downloading a file or a complete directory to Client (and saving directory/file as such on the client machine)

In the Navi/Swap page, the user can navigate to and double-click on items in the Mock Repository to populate a download file name text field and click a “Download” button to download files to the current User Machine Directory subdirectory, relative to that displayed in the GUI. File downloading is implemented on the server side, since the user would not have access to the Mock Repo machine directory directly, since they are supposed to be on different machines.

- Support sending the built test requests from the GUI/client to the server

Test requests are specified and built in the GUI and sent as messages to the server (through a separate user process).

- Support having the server save the named, time-date stamped test log to the Mock Repository

The server keeps track of how many Test Log Files it has created with a unique integer ID for each, and names Test Log Files accordingly. It’s Child Tester Processes save these to the MockRepo/TestLogs directory.

- Support receiving back a test status msg from the server

In the Statuses page, the user can browse Status Messages which contain the Test Driver and Tested Code Element names, the test number (ID), and the name of the Test Log File associated with the given test request. (Note: The notion of pass fail status is relative to each Test Driver / Tested Code Element pair, and since their may be numerous of these in each test request, the user is left to examine the Test Log File which contains these details).

On completion of the processing of each test request, the Child Tester Process which executed the test sends a named, time-date stamped test status message to the GUI. The message contains the name of the Test Log File associated with each given test request.

- Support allowing users to enter the system with a user name and password

User enters the system through the Log In / Log Out page with unique username/password. Any user with valid credentials may utilize the functionality of the Build And Test Page, the Statuses Page, and the Navi/Swap page.

- Support allowing only users with administrator privileges to create new users.

Only users with administrative privileges may create new users. This is done in the Log In/Log Out page.

Improper Usage Which May Cause the Program to Crash:

* It is the user’s responsibility to match Test Driver Elements with associated Tested Code Elements. If test driver elements are matched with un-associated tested code elements (or any other files, which it is possible to do), the program may crash. For example, matching MathTestDriver.dll with Math.dll may cause crash, since Math.dll defines an interface, not an implementation (I.e., it is not a tested code element).
* Double clicking the ListBoxes outside of a ListBoxItem may cause the program to crash.
* Program may crash if we revert (by altering source code) to using program 3 functions which send test requests but do not account for msg attribute extraction for attributes which aren't present. (Functions such as sendTenTimesNTestRequests(), for example, may do this if called, but these were left in without being called for use in future development).
* Generally, message passing synchronization (by design, not enforced), which is what the communication structure is build on, is not necessarily safe. In particular, if the user is able to click different buttons fast enough, this may cause message mis-matching or file transfer block-messaging overlap and crash the program.

Known Issues / Items which should be Implemented in Future Versions:

* A proper shutdown should be implemented, joining all threads and stopping all Comms.
* Should ensure proper shutdown on GUI program exit or Server program exit.
* The “Stop Server” button just causes the server to save Test Log File integer ID (test request count) to file, but this is implemented elsewhere on a per test basis. The “Stop Server Button” does nothing else but should shut down all the Comms and join all the threads.

###

* User must restart the server on each instance of the GUI program. (It might also be necessary that the server be running before the GUI process is started.) Still, Log File count (test request count), Log File integrity and user credentials are maintained between runs.

###

* No display for invalid credentials, access simply won’t be granted.
* User name safety not guaranteed (it may be possible for these to be over-written).
* The “Create New User” button requires two clicks after first logging in.
* Credential fields are not guaranteed to be cleared on log-in / log-out.
* Allow Administrators to grant administrator privileges to existing users.

###

* Use better versioning naming scheme.
* Associate versioning with dates for maintenance history.

###

* No file integrity safety guaranteed with uploading / downloading same-named files between the Mock Repo and the User Machine Directory.
* The project 3 sendFilePath and saveFilePath variables are dynamically modified, and their values should be verified to be reset to their original values (though this would only be relevant if source code is altered to utilize Project 3 functionality which is not used in Project 4).
* When in the Navi/Swap page or Build And Test pages, any file additions to directories are only reflected after navigating away from and then back to that directory.
* File transfers should be synchronized.
* Choose to output test results to console / test log file / or both.
* Implement a pop-up window to display test logs or other files.

###

* Note, new bug in version p312. For some reason, spawn(3) causing an error.

###

* GUI generated test request strings can only be created to contain one test driver element and one or more tested code elements (together, these constitute a single Test Element). It would be nice to be able to string together multiple Test Elements to send from the GUI, since the server was designed to handle strings composed of arbitrarily numerous Test Elements.
* ###
* Message relay redundancy (why create a new message when one can be redirected?).