ReqEx Test Plan Outline

Prepared by:

Zachary Bruggen
Nicholas Epler
Ivan Hernandez
Thomas Morrison

Faculty Advisor:

Dr. Khaled Slhoub

1. Test Plan Identifiers

Version	Author(s)	Reason for Change	Date Completed
1.0	Z. Bruggen, N. Epler, I. Hernandez, T. Morrison	Initial Writeup	10/4/2021

2. References

2.1 Project Plan [Link]

2.2 Requirements Document [Link]

3. Introductions

The purpose of this document is to outline how testing for the ReqEx system is going to be done. This master test plan outlines which features are going to be tested, and with what inputs test cases will be generated. With these test cases, an expected outcome will be created by the testers, and the actual results will be compared to those expected outcomes to determine if a test is successful or not.

4. Test Items

T1. Upload Input Files
T2. Display Uploaded Files
T3. Select and Navigate Through Files
T4. Generate Requirements Report
T5. Print Generated Report
T6. Save Generated Report
T7. Modify Generated Report

5. Software Risk Issues

The major risk in testing this software is the implementation of a new feature. If the features are not implemented within the schedule, it causes a high risk on the remainder of the testing schedule. A late implementation will push back the testing of the functionality, which in turn pushes back the accuracy testing.

The next major risk is changes to the requirements during the development. Again, this problem could occur often during the beginning of the project, but poses a great risk to the testing schedule, and the accuracy of the software.

Finally, the duration of the time limits between the project's milestones are tight enough to pose a risk to the testing schedule, and by extent the quality of the software as a whole.

6. Features to be Tested

Test Case	Section	How to Test	Expected Output	Priority
T 1.1	**2.2.1.1 -** Upload Input Files: Accept HLL source code	Select and upload the HLL source code.	File is successfully uploaded and the user receives confirmation.	High
T 1.2	**2.2.1.2 -** Upload Input Files: Accept Header File	Select and upload a Header File.	File is successfully uploaded and the user receives confirmation.	High
T 1.3	**2.2.1.3 -** Upload Input Files: Accept Text File	Select and upload a Text File.	File is successfully uploaded and the user receives confirmation.	High
T 1.4	**2.2.1.4 -** Upload Input Files: Upload multiple Files	Select and upload multiple files that the system accepts.	Multiple files are successfully uploaded and the user receives confirmation	Medium
T 1.5	**2.2.1.5 -** Upload Input Files: Upload a Folder	Select and upload a folder.	Folder is successfully uploaded and the user receives confirmation	Medium
T 2.1	**2.2.2.1 -** Display Previously Uploaded Files: Display uploaded Files	Upload a file.	File is successfully uploaded and displayed onto the screen.	High
T 2.2	**2.2.2.2 -** Display Previously Uploaded Files: Display uploaded Folders	Upload a folder(s) containing source code, header files, and/or text files.	Folder is successfully uploaded and displayed onto the screen.	Medium
T 3.1	**2.2.3.1 -** Select and Navigate Through Files: Navigate through list of	Click on the folder on the side panel.	Folder successfully displays the list of available inner folders and files.	High

	files			
T 3.2	**2.2.3.2 -** Select and Navigate Through Files: Select file to display	Click on the folder on the side panel, then click on a file that is not already being displayed	The File being currently displayed successfully changes to the file selected.	High
T 4.1	**2.2.4.1 -** Generate Requirements Report: Generate requirements for the whole program	Upload a folder containing all of the source code files, header files, and documentation of a program.	The system successfully generates an RER for the whole program.	High
T 4.2	**2.2.4.2 -** Generate Requirements Report: Generate requirements for multiple files	Upload several files to be analyzed.	The system successfully generates an RER for the files submitted.	Medium
T 4.3	**2.2.4.3 -** Generate Requirements Report: Generate requirements for a method/function	Upload a source code file that only has a single method to be analyzed.	The system successfully generates an RER for that single method.	High
T 4.4	**2.2.4.4 -** Generate Requirements Report: Generate requirements from source code	Upload a source code file	The system successfully generates an RER for the file.	High
T 4.5	**2.2.4.5 -** Generate Requirements Report: Generate requirements from a text file	Upload a text file	The system successfully generates an RER for the file.	Medium
T 4.6	**2.2.4.6 -** Generate Requirements Report: Generates requirements from a header file	Upload a header file	The system successfully generates an RER for the file.	Low
T 4.7	**2.2.4.7 -** Generate Requirements Report: Generates a simplified or a detailed report	Select the generate detailed report option. Select the generate simplified report option.	The system successfully displays the generated detailed RER or simplified RER.	High
T 5.1	**2.2.5.1 -** Print Generated Report:	Select the generate detailed report	The system successfully displays the generated	High

	Print detailed report	option.	detailed RER.	
T 5.2	**2.2.5.2 -** Print Generated Report: Print simplified report	Select the generate simplified report option.	The system successfully displays the generated simplified RER.	High
T 6.1	**2.2.6.1 -** Save Generated Report: Display multiple save formats	Click on the save report button.	The system successfully displays all of the available saving formats.	High
T 6.2	**2.2.6.2 -** Save Generated Report: Save report in a specific format	Click the save button and then select the format the report should be saved as.	The RER successfully saved to whatever format was selected.	High
T 6.3	**2.2.6.3 -** Save Generated Report: Select location to save to	Click the save button and then select the format the report should be saved as. Then specify the location to save to.	The RER is successfully saved to whatever directory that was specified.	High
T 7.1	**2.2.7.1 -** Modify Generated Report: Highlight text from report	Select text from the report and then click the highlighter button.	Text select has successfully changed to a bright yellow background.	Low
T 7.2	**2.2.7.2-** Modify Generated Report: Add text from report	Click on a section of the report and use the keyboard to send input.	Text successfully is added to the RER that was not successfully generated.	Medium
T7.3	**2.2.7.3-** Modify Generated Report: Delete text from report	Select a portion of the text from the report and press the backspace key.	Text from the RER is successfully deleted.	Medium

Reference the Requirements Document

7. Features not to be Tested

Currently, the feature to accept multiple HLLs will not be tested within this current version. The testing plan is to ensure that the software will successfully generate an RER for one HLL, and then implement the ability to utilize the software for other languages. This will come in a later version of the product, but will follow the same general guidelines described in this plan

8. Approach

The general approach for testing this system is to test functionality and accuracy of each feature as they are being implemented. During the beginning of a features implementation, testing will be done to ensure that the feature can operate alone, and then can operate within the system as a whole without causing a system failure. Then, once the testers have verified the feature is operating within the system, features can be tested for accuracy. Because the system is generating requirements from text analysis, there is a certain level of accuracy that the system is generating a passable output. Testers will perform accuracy tests multiple times to ensure that the system is generating a proper output that is within the limits of error.

9. Item Pass/Fail Criteria

Testing success is going to be determined by two different sets of criteria. Each particular feature is going to be tested alone, and then after implementation to ensure that it integrates into the program correctly. In terms of success or failure, a feature is going to be deemed successful if the feature generates the expected output according to its test case. If the implementation causes a system crash, or any other output besides the expected output is generated, the feature is going to be deemed a failure and testing will be suspended to properly implement it.

Once the feature is correctly implemented, the accuracy of certain features will be tested, namely the requirements generated. We cannot expect the system to match the expected requirement every time, so we will need to determine if the generated requirement is within the scope of the expected output. This sort of test will be done multiple times to determine the level of error that is within the system. If this level of error is within our limits, then the feature will be deemed as correct, and the test will pass.

10. Suspension Criteria and Resumption Requirements

If a new feature cannot be properly implemented, as indicated by the result of the test cases, then testing will be suspended. From there, the new feature and the system will be analyzed to determine the fault(s) that is preventing the new feature from being implemented. Once this fault is fixed, the testing plan for that particular feature can resume.

11. Test Deliverables

This document is acting as the master test plan, so all testing is going to be done based on the specifications laid out within this document, and it will be a deliverable to define the rationale behind the testing process. Following this strategy, test cases will be created by the test team in order to test individual features, and the case following its execution will act as a deliverable. Additionally, the features will be tested using samples of files that conform to the

requirements for each test case, so those samples will be deliverables in order for anyone to repeat the tests at their discretion.

12. Remaining Test Tasks

Every test item defined will be completed within the scope of this project. There will be no other test tasks that remain after the duration of the project.

13. Environmental Needs

Sample input files must follow the assumptions on the input that are outlined in the Requirements document linked above.

14. Staffing and Training Needs

The creators of this product are expected to fully understand how each individual component of the software will work. Therefore, there will be no significant training requirements in order to perform testing on this system. Additionally, each member will be included in the testing team, and will work together to test all features of the current version of the product.

15. Responsibilities

Responsibility will be shared equally between the creators of the product. As members of the team implement new features into the software, the team will decide which features take priority in testing. Each member will then be responsible for communicating with each other in order to ensure that testing is completed within the allocated time frame. Additionally, all members of the team will determine what constitutes between passing and failing any particular test.

16. Schedule

Preliminary testing will be conducted whenever a new feature is to be added. Testers will perform basic testing before implementation to ensure that the feature can operate in a standalone fashion, and then perform stronger testing to ensure that the feature integerates correctly into the system as a whole.

17. Planning Risks and Contingencies

The risks associated with the testing process are the late delivery of the software, changes to the original requirements that were made during development, and lack of personnel resources for the project.

To combat these risks, the testing team will decide between any number of the following actions to take in order to provide a deliverable product.

- Decrease the number of tests performed
    - This will done by lowering the tests for features with low priority first
- Increase the number of acceptable defects
    - This will done by increasing the limits of error on low priority features
- The test team will work beyond normal working hours
- Push back the testing schedule an appropriate amount of days
    - This should be a last resort, because the milestones must be met by the date
- The scope of the plan or the project will be changed
    - This should be the absolute last resort, as the project scope should only be changed when nothing else can be done

18. Approvals

All members of the team will be able to dictate when testing is completed, and the project can move on to the next phase. As the team performs the testing process, they will discuss with each other whether more testing needs to be performed on any particular aspect of the software.

19. Glossary

HLL	High Level Language (C, C++, Java, etc.)
RER	Requirements Extraction Report