**University of Houston Clear Lake**

Spring ‘22

CSCI 4388.02

Senior Project in Computer Science

**Test Plan Document**



The Data Extractors

04/13/2022

**Test Plan**

Purpose

The Word Text Data Extractor is a piece of software made in Python that will take Microsoft Word documents, mainly instruction manuals or documents with procedures for the reader to follow, with the ‘.docx’ file extension in as input. The Extractor will then read through the .docx file, identify the features of the document (tables, lists, headers and footers, etc.), and then finally display any text information that could be deemed important/relevant via an XML schema.

Features to be tested

Base States:

BS-0: No applications running.

BS-1: Base Command Line Terminal

BS-2: Command Line Terminal with Program Running & Menu Prompt

BS-3: Input Filepath Specified & Valid

BS-4: Output Location Specified & Valid

BS-5: Output File Name Specified & Valid

BS-6: Input and Output Filepaths are Specified & Valid

BS-7: Help Menu

| Name | BS-0: No applications running. |
| --- | --- |
| Requirement | N/A |
| Preconditions | * User is running a machine with Windows OS and a command line terminal. * User is logged in to Windows. |
| Steps | 1. Log in to Windows   Or   1. User has closed the Terminal window. |
| Expected Results | User is logged in to the Windows OS.  No relevant applications are running. |

| Name | BS-1: Base Command Line Terminal |
| --- | --- |
| Requirement | N/A |
| Preconditions | * Base State-0 |
| Steps | 1. Launch “Command Prompt” from the Windows Start menu. 2. User’s decision |
| Expected Results | Empty Command Line Terminal window is open and awaiting further instruction. |

| Name | BS-2: Command Line Terminal with Program Running & Menu Prompt |
| --- | --- |
| Requirement | N/A |
| Preconditions | * User is running a machine with Windows OS and a command line terminal. No applications are running. (Base State-0) |
| Steps | 1. Launch “Command Prompt” from the Windows Start menu, or similar Command Line Terminal. 2. Launch the Data Extractor program. 3. User’s decision |
| Expected Results | Command Line Terminal window is open and displaying the program’s command menu.  Program is awaiting further instruction. |

| Name | BS-3: Input Filepath Specified & Valid |
| --- | --- |
| Requirement | N/A |
| Preconditions | * Base State-2 * Docx format document is present on filesystem. * User has used the input command, and provided a valid file path. |
| Steps | 1. User has launched a CLT 2. User has started the program (BS-2) 3. User enters the input command, followed by a valid file path. 4. User’s decision |
| Expected Results | The program has validated the specified input file path.  Program is awaiting further instruction. |

| Name | BS-4: Output Location Specified & Valid |
| --- | --- |
| Requirement | N/A |
| Preconditions | * BS-2 * Output location exists in filesystem * Input document has been validated (optional) * Input document has been parsed (optional) |
| Steps | 1. User has reached Base State-2 2. User enters outloc command 3. User enters a valid directory location within the filesystem. 4. User’s decision |
| Expected Results | The program has validated the specified directory.  Program is awaiting further instruction. |

| Name | BS-5: Output File Name Specified & Valid |
| --- | --- |
| Requirement | N/A |
| Preconditions | * BS-2 * User’s desired file name is possible. (No invalid characters) * Input document has been validated (optional) * Input document has been parsed (optional) |
| Steps | 1. User has reached Base State-2 2. User enters outname command 3. User enters a valid name for a file 4. User’s decision |
| Expected Results | The program has verified the filename is valid and available.  Program is awaiting further instruction. |

| Name | BS-6: Input and Output Filepaths are Specified & Valid |
| --- | --- |
| Requirement | N/A |
| Preconditions | * The user has reached BS-2 * The user has used the input command properly (BS-3) * User has used the outloc command properly (BS-4) * User has used the outname command properly (BS-5) |
| Steps | 1. User has previously used the input command 2. User has previously used the outloc and outname commands 3. User’s decision |
| Expected Results | Program has validated the user specified input file path.  Program has validated the user specified output file path.  Program is awaiting further instruction. |

| Name | BS-7: Help Menu |
| --- | --- |
| Requirement | N/A |
| Preconditions | * Base State-2 * User has entered the Help command |
| Steps | 1. User inputs the command “help” 2. Program displays help menu 3. User’s decision |
| Expected Results | Program displays the help menu.  Program is awaiting further discussion. |

Test Cases:

TC-1: Parse and Label Docx Document (UC-1)

TC-2: Set Output File Location (UC-2)

TC-3: Set Output File Name (UC-3)

TC-4: Set Input File Location and Name (UC-4)

TC-5: Use Help Options (UC-5)

TC-6: Start the Program (UC-6)

TC-7: End the Program (UC-7)

| Name | (TC-1) Verify Functionality of the Parse and Labeling Feature |
| --- | --- |
| Requirement | UC-1 (Parse and Label docx Document) |
| Preconditions | * Software is waiting in the main menu for the input command (BS-2). * Document is on the filesystem and ready. * Input location and filename has been specified and verified. * Output location has been specified and verified. * In Base State-6 |
| Steps | 1. The user enters the parse command that will perform this functionality into the command line terminal. 2. The output file will follow a schema that describes how different parts of the text will be labeled. 3. The output file is created in the specified directory in their filesystem with the specified name. |
| Expected Results | 1. There will be a properly parsed and label XML file saved in the specified location, with the specified name. 2. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed. (BS-2) |

| Name | (TC-2) Verify that the outloc command successfully sets the save location of the XML output |
| --- | --- |
| Requirement | UC-2 (Set Output File Location) |
| Preconditions | * Software is waiting in the main menu for the input command (BS-2). * Directory should exist on the system |
| Steps | 1. Enter outloc command into the terminal 2. Enter the file path to the directory where the output file will be saved (BS-4) |
| Expected Results | 1. If path does not exist the user will be notified by a message and will proceed to ask the user to input another path. 2. If the directory exists, the program will display a message confirming that the output location has been set. (BS-4) 3. Program will then return back to the main menu (BS-2). |

| Name | (TC-3) Verify that the outname command successfully sets the name of the XML output |
| --- | --- |
| Requirement | UC-3 (Set Output File Name) |
| Preconditions | * Software is waiting in the main menu for the input command (BS-2). * The outloc command has been previously executed successfully (BS-5) |
| Steps | 1. Enter outname command into the terminal 2. Enter the name of the XML output file |
| Expected Results | 1. The name entered should be checked for any illegal characters. 2. If illegal characters are found, users will be asked to input the name again. 3. If no illegal characters are found the program will display a message confirming that the XML output file name has been set. (BS-5) 4. Program will then return back to the main menu(BS-2). |

| Name | (TC-4) Verify That the input command successfully reads a file path input |
| --- | --- |
| Requirement | UC-4 (Set Input File Location and Name ) |
| Preconditions | * Software is waiting in the main menu for the input command (BS-2). * File exists on the system |
| Steps | 1. Enter the input command. 2. Enter the absolute file path to the file. (BS-3) |
| Expected Results | 1. The software should check if the file path exists 2. If it does not exists the user should be prompted to enter another file 3. If the file does exist a message should be displayed indicating the file was loaded successfully. (BS-3) 4. Program will then return back to the main menu (BS-2). |

| Name | (TC-5) Verify that the help menu opens and displays correct information |
| --- | --- |
| Requirement | UC-5 (Help) |
| Preconditions | * Software should be in the main menu(BS-2) |
| Steps | 1. Enter the help command 2. Help menu is opened (BS-7) 3. Enter help followed by the command they want to see help information. |
| Expected Results | 1. Help information for specified command is displayed 2. Software returns to the main help menu waiting to receive another help command (BS-7) 3. If user enters exit the software will return to the main menu (BS-2) |

| Name | (TC-6) Verify that the program successfully starts |
| --- | --- |
| Requirement | UC-6 (Start the Program) |
| Preconditions | * User is running a machine with Windows OS and a command line terminal. No applications are running. (BS-0) |
| Steps | 1. Launch “Command Prompt” from the Windows Start menu, or Python IDLE. (BS-1) 2. Run the Data Extractor program. |
| Expected Results | 1. Program should start successfully and display the main menu (BS-2) |

| Name | (TC-7) Verify that the program successfully terminates |
| --- | --- |
| Requirement | UC-7 (End the Program) |
| Preconditions | * The software should be running from a python command line terminal available to receive input with a menu displayed. (BS-2) |
| Steps | 1. Enter exit command |
| Expected Results | 1. The program is terminated. |

Features not to be tested

Because of the nature of our project, there are no features of the program that are not included in the testing.

Each command will have to be tested, which will involve each method and/or function being tested.

Approach

We will utilize a combination of testing techniques for our program. On the functional testing side we will, and have been, unit testing the different features of the program as we develop them. We also perform integration testing once different cooperating pieces of the program are complete. When one relies upon another, we test that relationship. We will be utilizing system testing to make sure the entire program functions as it should once we complete all coding.

After these testing phases we will include Mr. Tang from Tietronix to ensure a proper Acceptance Testing phase is completed before finalizing our project. Part of this acceptance testing phase, we may include a non-function performance test just to ensure its performance is up to Mr. Tang’s approval.

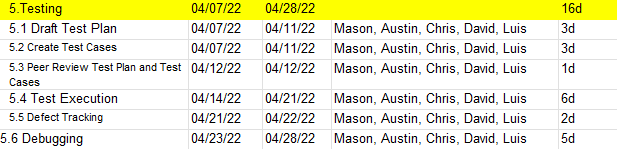
Suspension criteria and resumption requirements

| Suspension Criteria | Resumption Requirements |
| --- | --- |
| Commands are not working as intended. | The command must be fixed immediately due to some commands being dependent on others. Once the team resolves the issue, return to testing. |
| Significant errors in the XML output of the Word Document | Team will meet virtually to solve the error to resume testing. |
| Any of the official Test Cases fail | Resolve the failure, ensure all official cases pass |

Environmental Needs

The basic requirements for this program are a modern computer or similar device that is capable of running Windows 10 with Microsoft Office, as well as small python programs.

Schedule



Acceptance criteria

Stakeholder sign-off and consensus of satisfaction between all team members. The program should run without defects, with no environmental error. Unit Tests should be completed without error, as well as integration and system testing should produce no errors. Only risk of error could occur from context confusion within the document itself, or disordering of the output XML schema.

Roles and responsibilities

Mason Lanham - Project Test Lead, Testing Manager, Test Approver, Tester, Debugger, Reviewer.

Austin Meredith - Testing Manager, Test Designer, Test Approver, Tester, Debugger, Reviewer.

Luis Carrillo - Test Approver, Tester, Debugger, Reviewer.

Chris Mendoza - Testing Manager, Test Designer, Test Approver, Tester, Debugger, Reviewer.

David Garcia - Test Approver, Tester, Debugger, Reviewer.

Descriptions:

Project Test Lead – This is the person responsible for oversight of testing on the project.

Testing Manager – This is the person accountable for conducting quality assurance testing and executing on the test plan.

Test Designer – This is the person responsible for creating the test scripts, scenarios, test lives, and so on that make up the tests to be performed.

Test Approver – This is the person responsible for reviewing, validating, and approving the test materials created by the Test Designer.

Tester – This is the person responsible for executing the test scripts, and reporting the results.

Debugger - Each developer has the role of debugger, as we fix our code as we work.

Reviewer – This is the person responsible for reviewing reports from the testers and determining what subsequent actions will be taken.