**University of Houston Clear Lake**

Spring ‘22

CSCI 4388.02

Senior Project in Computer Science

**Software Requirements Specification**



The Data Extractors

02/16/2022

**Data Extractor Software Requirements Specification**

1. **Introduction**
2. ***Purpose***

This document will function as a guide for the software developers who will be responsible for designing, developing, and testing the Word Data Text Extractor project. This ‘Requirements Specification Document’ has been written to give the developers involved in this project a clear and concise overview of their responsibilities while working on this project.

1. ***Scope***

In order to give the reader a clear understanding of our Word Text Data Extractor project, this document consists of descriptions of the use cases, functional requirements, nonfunctional requirements, and overall functionality of our software.

1. ***System Overview***

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.

For example, let’s say an employee safety manual for the company you work for was input into the Extractor. This manual includes general information about the company, images of emergency escape routes, and a bulleted list of safety rules that all employees must follow. In this case, the Extractor would read through the entire document, extract and save any images as separate files, and then display all of the important text information, which, in this case, would be the bulleted list of safety rules.

1. ***References***

Openspecs-Office. “[MS-Docx]: Word Extensions to the Office Open XML (.Docx) File Format.” *Microsoft Docs*, Microsoft, https://docs.microsoft.com/en-us/openspecs/office\_standards/ms-docx/b839fe1f-e1ca-4fa6-8c26-5954d0abbccd.

Python “doc-x” library documentation. (2013). Read The Docs. <https://python-docx.readthedocs.io/en/latest/>

1. **Definitions**

***Microsoft Word*** - Word processing software developed and sold by Microsoft.

***.docx*** - The modern file format for Microsoft Word documents. Introduced in 2007, it replaced the .doc file format.

***Python*** - High-level, open-source programming language that can be used for a wide array of purposes. Designed to be easy to read and simple to learn.

***‘Python-docx’*** - Python library which can be used to create and manipulate Microsoft Word documents that have the .docx file format.

***XML*** - Stands for Extensible Markup Language. Used to create formats which can store data in a way that can be easily read, stored, and shared.

***Parser*** - Software component that takes input data (usually text) and builds a data structure

1. **Use Cases**

**UC-1: Parse and Label Document**

Summary

This use case describes how the user would pull all the text and other data from a word document and save it in an output file that follows an XML schema with labels for different parts of the text. The labels will represent if the text is a section title, paragraph, part of a procedure, or belongs to a row of a table. This list can be expanded as needed. Some additional items, such as graphics and charts will be saved separately.

Rationale

This would be helpful for retrieving and parsing the document for use by the next tool.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.
2. The word file that you want to pull the text from should be available on your filesystem.
3. The output file location and name should already be specified.
4. The input file location and name should already be specified.

Basic Course of Events

1. The user enters the 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. This schema is described below.
3. Section titles are labeled as such.
4. Paragraphs will be labeled sequentially with their section title.
5. Parts of a procedure are labeled sequentially with their procedure title.
6. Table cells will be labeled by what they row and column they correspond to. If there is a label in the table that will be applied.
7. Graphics and charts will be saved in a separate file with a link to them in the output file.
8. The output file is created in the specified directory in their filesystem with the specified name.

Alternative Paths

1. If there is no name and location of the word document selected, then they will be prompted for one.
2. If there is no output file location specified, the user will be prompted to select one before the function is performed.
3. If there is no name specified for the output file, the user will be prompted to select one before the function is performed.

Postconditions

1. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed.

**UC-2: Set Output File Location**

Summary

This use case describes how the user would set the location in their file system for their output files.

Rationale

This allows the user to send their output files where they are needed for the next tool.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.
2. The directory that the user chooses should exist on your file system.

Basic Course of Events

1. The user enters the command to set the output file location into the command line terminal with an argument specifying which directory to set as the output file directory.
2. The output file location is specified for the parse and label document command to use.

Alternative Paths

1. If the directory specified doesn’t exist, then the user will be informed of the error.

Postconditions

1. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed.

**UC-3: Set Output File Name**

Summary

This use case describes how the user would set the name of their output file.

Rationale

This user will need to be able to set what the name of their output file will be for it to be used by the next tool.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.
2. The directory that the user chooses should exist on your file system.

Basic Course of Events

1. The user enters the command that will set the output file name into the command line terminal with an argument specifying the name of the file.
2. The output file name is specified for the parse and label document command to use.

Alternative Paths

1. If the directory specified doesn’t exist, then the user will be informed of the error.

Postconditions

1. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed.

**UC-4: Set Input File Location and Name**

Summary

This use case describes how the user would set the location in their file system for their input file.

Rationale

This allows the user to select the input file they will be working on.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.
2. The directory and file that the user chooses should exist on your file system.

Basic Course of Events

1. Software prompts the user for a command.
2. The user enters a command to open a specific directory/location where the .docx file is located.
3. The software displays which directory/location it has now opened.

Alternative Paths

1. If the directory or file specified doesn’t exist, then the user will be informed of the error.

Postconditions

1. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed.

**UC-5: Help**

Summary

This use case describes how the user would get help with understanding how to use our software.

Rationale

Having this functionality will make our software easier to use.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.

Basic Course of Events

1. The user will enter the help command.
2. If the help command is used without arguments, it will explain how to use the help command.
3. If the help command is used with a command as an argument it will explain how that command works, and acceptable data to use as arguments for that command.

Alternative Paths

None

Postconditions

1. The software will return to running from the python command line terminal and will be available to receive input with the menu displayed.

**UC-6: Start the Program**

Summary

This use case describes how the user would start the program.

Rationale

To be able to use the program it needs to be started.

Users

All users would be able to use this functionality.

Preconditions

1. A Python command line of version 3.3 or newer is installed on your system.

Basic Course of Events

1. The user opens a python command line terminal.
2. The user types the command to start the program.
3. The program starts.

Alternative Paths

1. The program could be started from an IDE such as IDLE.
2. The program could be started from a different command line terminal.

Postconditions

1. The software will run from the python command line terminal and will be available to receive input with the menu displayed.

**UC-7: End the Program**

Summary

This use case describes how the user would end the execution of the program.

Rationale

The program will need to end execution when the user is done.

Users

All users would be able to use this functionality.

Preconditions

1. The software should be running from a python command line terminal available to receive input with a menu displayed.

Basic Course of Events

1. The user enters the command to exit the program.
2. The program stops execution and the user returns to the base command line terminal.

Alternative Paths

1. The user ends the program by exiting the command line terminal.
2. The user ends the program by using a service on their operating system.

Postconditions

The program is terminated.

1. **Functional Requirements**

| Name | FR-1: Command Line Input |
| --- | --- |
| Summary | The user should be able to input the file using a command line |
| Rationale | Users need a way to select the word document they need. |
| Requirements | * Program needs to allow the user to input the word document for data extraction using a command line prompt. User will input the directory and filename |
| References | UC-4 Set Input File Location and Name |

| Name | FR-2: Parsing of Word Document |
| --- | --- |
| Summary | The program should be able to read the data of the document |
| Rationale | Program can’t do its main task without being able to read the document |
| Requirements | * Program should parse the word document input to read the text that is in the file. |
| References | UC-1 Parse and Label Document |

| Name | FR-3: XML File Output |
| --- | --- |
| Summary | The program should output the data within the word document input onto a XMLfile. |
| Rationale | User needs to have a way of seeing the extracted data. |
| Requirements | * The data that is extracted needs to be put into an XML file. * Graphics and Images should be placed into a separate file that references the XML file. |
| References | UC-2 Set Output File Location  UC-3: Set Output File Name |

| Name | FR-4:Accuracy |
| --- | --- |
| Summary | Program should extract the data correctly from the word document |
| Rationale | If the program extracts data incorrectly users will have no reason to use it. |
| Requirements | * Program will correctly retrieve the text and properly label them. * The data in the XML File output should match the data in the word document |
| References | UC-1 Parse and Label Document |

1. **Nonfunctional Requirements**

| Name | NF-1:Performance constraints for retrieving the data. |
| --- | --- |
| Summary | Program should extract data quickly. |
| Rationale | If the extraction is not quick and efficient this will slow down the productivity of the user/company that is using the program |
| Requirements | * Data should be extracted in under 30 seconds on a document that is 50 pages or less |
| References | UC-1 Parse and Label document |

| Name | NF-2:Reliability |
| --- | --- |
| Summary | Program should work with minimal failures. |
| Rationale | Failures could frustrate the user and limit the work they could do with the program. |
| Requirements | * Program should have a less than 5% failure rate when parsing files. * Program should never fail wherever completing functions other than parsing files.(Input and Output) |
| References |  |

| Name | NF-3:Usability |
| --- | --- |
| Summary | Program should be easy to use. |
| Rationale | Nobody wants to work using something that makes the task even more difficult |
| Requirements | * Program should be user friendly by displaying instructions clearly. * Anyone that has used a command prompt should be able to use it. * The command prompt is clear of unnecessary elements and content to not confuse the user. * Program will include a help option in case the user gets stuck and needs further instructions. |
| References | UC-5 Help |

| Name | NF-4: Portability |
| --- | --- |
| Summary | Program should be portable to different systems. |
| Rationale | The program will be available to more users if it is portable. |
| Requirements | * The program should be portable to systems with Python 3.3 or newer installed. |
| References | Dependencies: https://python-docx.readthedocs.io/en/latest/user/install.html |

| Name | NF-5: Ethics |
| --- | --- |
| Summary | Program should not violate the access rules of Microsoft Word. |
| Rationale | The tool we are making is not intended to be used for purposes violating computer ethics. |
| Requirements | * Program should only be able to extract data if the user has access to the document they will be extracting. * Program should not be easily modifiable to access rules to be worked around. |
| References | Protected Documents: https://support.microsoft.com/en-us/office/protect-a-document-with-a-password-05084cc3-300d-4c1a-8416-38d3e37d6826 |

| Name | NF-6: Data Integrity |
| --- | --- |
| Summary | Program will follow specific XML schemas to capture the meaning of the data it records. |
| Rationale | The intention of this project is to extract relevant data from word documents. |
| Requirements | * The output file will follow rules of the XML schema selected. * The meaning of the extracted data will not change depending on which command is used. |
| References | UC-1: Parse and Label Document |

| Name | NF-7: Compatibility |
| --- | --- |
| Summary | Program will be compatible with only specific types of word documents. |
| Rationale | There is no need to develop for types of documents that are no longer being used. |
| Requirements | * Program is only compatible with files in the .docx format. Specifically the XML-based file format for Word 2007 and newer. |
| References | File Format Reference: https://docs.microsoft.com/en-us/deployoffice/compat/office-file-format-reference |

| Name | NF-8: Robustness |
| --- | --- |
| Summary | Program will be capable of gracefully handling user input errors, or unexpected contents in a document. |
| Rationale | The program unexpectedly crashing, or being unable to handle some contents of a document could result in frustrating experiences for users. |
| Requirements | * User input errors will result in their input being rejected, and them being prompted to enter the input again. * Unexpected document contents will be ignored, except to inform the user of its presence. * Corrupted Documents will not have their data extracted, but users will be informed of their status. |
| References | UC-1: Parse and Label Document |