Requirements Tracing/Management Software

Adrian Bernardino, Jan William Haug, Stephania Rey

Major in Software Engineering

Course Instructor: Simon Fan Faculty Advisor: Asif Imran

Industry Sponsor: Tandem Diabetes Care Project Mentor: Tom Ulrich

A capstone project report submitted to the faculty of Computer Science and Information Systems, California State University, San Marcos

May 2023

(Version 5.0)

Technical Report Series: CSU-SM-CSIS-Class2023-Sec-001-Team-006

Table of Contents

1.0 Abstract ……Pg. 7-8

1.1 Project Goals ……Pg. 9

1.2 Potential Impacts ……Pg.9

2.0 Report Revision History ……Pg. 9

2.1 Changes in Version 1.0…… Pg.9

2.1.1 Changes in Version 1.5……Pg.10

2.1.2 Changes in Version 2.0……Pg.10

2.1.3 Changes in Version 2.5……Pg.10

2.1.4 Changes in Version 3.0……Pg.10-11

2.1.5 Changes in Version 3.5……Pg. 11

2.16 Changes in Version 4.0……Pg. 11

2.17 Changes in Version 5.0…… Pg. 11-12

3.0 Problem Statement …… Pg. 12

3.1 Business Background…… Pg.12

3.2 Needs(why)…… Pg. 12-13

3.3 Objectives(what)……Pg. 13

3.4 Lessons Learned ……Pg.13-14

4.0 Requirements……Pg. 14

4.1 User Requirement……Pg. 14

4.1.2 User Groups……Pg. 14

4.1.3 Functional Requirements(User)……Pg. 15

4.1.3.1 Project Scope……Pg.15-16

4.1.3.2 User Scenarios……Pg.17-18

4.1.3.3 User Functional Requirements ……Pg.18

4.1.4 Non-Functional Requirements ……Pg.19

4.1.4.1 Product: Usability Requirements……Pg.19

4.1.4.2 Organizational: Development Requirements……Pg.19

4.2.2 System Requirements……Pg. 19
4.2.1 Functional Requirements……Pg. 19
4.2.1.1 System Functional Requirements……Pg. 19-20
4.2.2 Non-Functional Requirements……Pg. 20
4.2.2.1 Product: Usability Requirements……Pg. 20
4.2.2.2 Organizational: Development Requirements……Pg.21
4.3 Requirements Trace Table……Pg. 21
4.4 Key Challenges (Requirements) ……Pg. 21
4.4.1 Compatibility Across operating Systems……Pg. 21
4.4.2 Optimizing Performance for Large Data Sets ……Pg.21-22
4.4.3 Integration with External Tools ……Pg.22
4.4.4 Ensuring Usability ……Pg.22
5.0 Exploratory Studies……Pg. 22
5.1 Relevant Development Frameworks…… Pg. 22
5.2 Relevant Solution Techniques……Pg. 22-23
5.3 Broader Impacts……Pg. 23-24
6.0 System Design……Pg.24
6.1 Architectural Design……Pg. 24-26
6.2 Structural Design……Pg. 27
6.3 User-Interface Design……Pg. 28
6.4 Behavioral Design……Pg.29-32
6.5 Design Alternatives & Design Rationale……Pg.33-34
6.6 Key Challenges(System Design) ……Pg.34
6.6.1 Balancing Flexibility and Complexity Key ……Pg.34
6.6.2 Achieving Platform Independence……Pg.35
6.6.3 Managing Interactions and Dependencies ……Pg.35
7.0 System Implementation……Pg. 35
7.1 Programming Languages and Tools…… ……Pg. 35
7.2 Coding Convention……Pg. 35

7.3 Code Version Control……Pg. 36-37
7.4 Justification for Chosen Approaches……Pg. 37
7.5 System Implementation ……Pg.33-34
7.6 Key Challenges (System Implementation)……Pg.34
7.6.1 Integration of Libraries and Technologies……Pg.34
7.6.2 Ensuring Code Quality and Maintainability ……Pg.34
7.6.3 Managing Code Versions and Dependencies……Pg.34
7.6.4 Testing and Debugging……Pg.34
8.0 Testing……Pg. 35
8.1 Types of Tests……Pg. 35-36
8.2 Test Cases……Pg. 36-37
8.3 Key Challenges (Testing)……Pg. 36-37
9.0 Challenges & Open Issues……Pg.38
9.1 Challenges Faced in Requirements Engineering……Pg.38
10.0 System Manuals……Pg. 39
10.1.1 How to set up development Environment……Pg. 39
11.0 System Testing……Pg. 40-42
12.0 References Appendix…… ……Pg.42
U (generated form Use Case Printout on Capstone)……Pg.42-43
Table 4.1. Use Case Index Table……Pg.42-43
Table 4.2. Use Case UC-001……Pg.43
Table 4.3. Use Case UC-002……Pg.44
Table 4.4. Use Case UC-003……Pg.44
Table 4.5. Use Case UC-004……Pg.45
Table 4.6. Use Case UC-005……Pg.45
Table 4.7. Use Case UC-006……Pg.46
Table 4.8. Use Case UC-007……Pg.46
R (generated from Requirements Printout on Capstone)……Pg.47
Table 4.9. User Functional Requirements: UF-A ……Pg.47

Table 4.10. User Functional Requirements: UF-B……..Pg.47

Table 4.11. User Functional Requirements: UF-C……..Pg.47

Table 4.12. User Functional Requirements: UF-D……..Pg.48

Table 4.13. User Functional Requirements: UF-E……..Pg.48

Table 4.14. User Functional Requirements: UF-F……..Pg.48

Table 4.15. User Functional Requirements: UF-G……..Pg.48

Table 4.16. User Functional Requirements: UF-H……..Pg.49

Table 4.17. User Non-Functional Requirements: UP-01……..Pg.49

Table 4.18. User Non-Functional Requirements: UO-01……..Pg.49

Table 4.19. System Functional Requirements: SF-A-01……..Pg.49

Table 4.20. System Functional Requirements: SF-B-01……..Pg.50

Table 4.21. System Functional Requirements: SF-B-02……..Pg.50

Table 4.22. System Functional Requirements: SF-B-03……..Pg.50

Table 4.23. System Functional Requirements: SF-C-01……..Pg.50

Table 4.24. System Functional Requirements: SF-D-01……..Pg.50-51

Table 4.25. System Functional Requirements: SF-E-01……..Pg.51

Table 4.26. System Functional Requirements: SF-F-01……..Pg.51

Table 4.27. System Functional Requirements: SF-G-01……..Pg.51

Table 4.28. System Functional Requirements: SF-H-01……..Pg.52

Table 4.29. System Non-Functional Requirements: SP-01-01……..Pg.52

Table 4.30. System Non-Functional Requirements: SO-01-01……..Pg.52

Table 4.3. Requirement Trace Table:……..Pg. 53

Table 4.31 Mapping from User Requirements to System Requirements……..Pg. 53

T (Code Source)……..Pg. 53

Main.py ……..Pg. 53

Targest.py……..Pg. 54-58

Targest2.py ……..Pg. 58-108

Gui.py ……..Pg. 108-115

TE System Testing (Test Case Design) ……..Pg.115

Table 8.2.1 Test Suite TS-001: Child Tags…….. Pg. 115

Table 8.2.2 Test Suite TS-002: Parent Tags…….Pg. 115

Table 8.2.1 Test Suite TS-001: Child Tags ……..Pg.115

Table 8.2.2 Test Suite TS-002: Parent Tags ……..Pg.115

Table 8.2.3 Test Suite TS-003: TC-003……..Pg.115

Table 8.2.4 Test Suite TS-004: Child Tags, with at least one parent ……..Pg.115

Table 8.2.5 Test Suite TS-005: TC-005……..Pg.115

Table 8.2.6 Test Case TC-001……..Pg.116

Table 8.2.7 Test Case TC-002……..Pg.116

Table 8.2.8 Test Case TC-003……..Pg.116

Table 8.2.9 Test Case TC-004……..Pg.117

Table 8.2.10 Test Case TC-005……..Pg.117

Table 8.3.1 Execution Report of Test Case TC-001……..Pg.118

Table 8.3.2 Execution Report of Test Case TC-002……..Pg.118

Table 8.3.3 Execution Report of Test Case TC-003……..Pg.119

Table 8.3.4 Execution Report of Test Case TC-004……..Pg.119

Table 8.3.5 Execution Report of Test Case TC-005……..Pg.119

Additional Test Cases……..Pg.123-124

12.1 Glossary of Relevant Domain Technology……..Pg.124

13.0 Reference Documents……..Pg.125-126

1.0 Abstract

Before TARGEST, writing software for life/flight/mission critical applications and managing requirements has always been a long process. All requirements had to be laid out in terms of software, design, specification, and testing. Each requirement had to be taken into account alongside the system consideration. In addition, every requirement had to be adequately tested to guarantee the smooth delivery of the product. Twenty years ago, there were commercially established requirements tracing software available on the market. However, they ran poorly when it came to requirements traceability due to inconsistencies with the software, running compatibility issues, an limitations as a result of the software not being updated regularly, managed and being prone to crashing. This is where (TARGEST) comes in to solve this problem. TARGEST stands for technical abstraction report generator extraction software tool. It is an application that runs on the sole purpose of conducting requirements document management. Our tool converts documents into a requirements document that will be used for code and testing purposes. Our tool enables engineers to find misunderstandings and help invent requirements to ensure smooth productivity. In addition our tool is used to demonstrate accuracy of the documents and sufficiency of test coverage for itself. Many companies need this kind of software, so we came with a solution for that. The value of TARGEST is that it is a lightweight alternative.

In the completed project, several essential features were implemented to create a robust and efficient system for managing and analyzing requirements in Word documents. The document parser, developed using the Python/Docx library, had been designed to accurately read and extract relevant paragraphs from the requirement documents by identifying and processing various tags, including parent requirement tags, child requirement tags, and TBV tags. A hierarchy structure was also created to represent the relationships between parent and child requirements, allowing for better organization and understanding of the interdependencies within the system. In addition, algorithms were devised to analyze the requirements hierarchy and generate various reports, such as identifying orphan tags, childless tags, and TBV tags. These reports were then exported in Excel format using the Python/ Xlwings library, offering valuable insights into the relationships and traceability of the requirements. Furthermore, the implemented algorithms enabled the visualization of the flow of requirements between different documents, effectively displaying these trees within the application's graphical user interface (GUI). By incorporating these key algorithms, the requirement tracing tool successfully managed and analyzed the relationships between requirements in Word documents, generating essential reports for validating the accuracy of the documents and requirements.

Our motivation behind this project is our partnerships with Tandem Diabetes Care, a local medical and technical based company that develops medical technologies such as insulin pumps. They were looking for a requirement tracing software that will extract information from documents, review it before proceeding to doing the tasks. In addition, every change that Tandem makes in the insulin pump needs approval by FDA. Our software will save Tandem a week’s worth of work and time. Our software will

accomplish this task by taking data from any provided documents, performs a search of tag relation, then extracts tags and content, once the extraction process has been completed, it regroups the selected information and reorganizes it in terms of the relationship between leading tags, trailing tags and orphan tags. This process usually is performed manually and take about a week's work to organize this content. Our software takes care of this. Through this partnership, our team created TARGEST.

The Requirements section details the system implementation process of the TARGEST software, considering the project requirements, programming languages and tools, coding conventions, version control, and the justification for chosen approaches. It highlights the iterative approach taken in the development process, outlining the key components of the application in order to meet the project requirements effectively. The functional requirements we focused on user and system requirementes to describe how the user or software can achieve user and system requirements. Functional requirements drive the application architecture of a system. Nonfunctional requirements include all the features we wanted to implement which involved requirements users and system engage with in terms of Product to determine the quality of services, Organizational to establish the software's development structure, and External which is factored in by safety or security, cultural or social, or political as well. Regarding utilizing different frameworks and tools, we decided to move with the tools that helped improve the functionality and user interface of our software. Tkinter we used for our web application primarily to simplfy the GUI creation. Open-source packages like Xlwings and libraries like Python-Docx and Pandas helped with the handling of data management and analysis. Finally regular expressions we resorted to to analyze the different text patterns found through the documents we were extracting. The overall architecture system design promotes loose coupling by separating the application into three interconnected components. Our Model represents the data and business logic of the application. The View displays data to the user and handles user interaction. In our Controller component, we have folders that serve a specific purpose for managing the flow of data between the Model and the View. The system implementation section outlines the programming languages, tools, coding conventions, and version control used in the development process, as well as the rationale behind the chosen approaches. It describes the iterative approach taken during the implementation, highlighting key components, such as GUI prototyping, requirements gathering, and the integration of components. For the System Testing section, the team initially conducted testing by running the software and generating reports, which revealed various bugs and flaws, such as missing or misplaced tags and incorrect linkages between tags. To improve the effectiveness of our testing, we implemented Py Unit testing, which allowed us to create targeted test cases for individual functions and improve the overall quality and reliability of the software.

1.1 Project Goals

TARGEST goal was to develop a software application that would be able to coordinate different documents. In doing so, we created and assign the software to conduct requirements traceability as it would derive information from existing requirements document and exhibit tags and their relationship to each other. As this process occurred, documents would be automatically produced instantly and that they graphically communicate the documents at their respective level.

1.2 Potential Impacts

The team focused on meeting the project initiatives first. Before starting creating the software, we made sure the initial requirements of the projects were being met, and laid out the specifics before starting on the code. We are made sure we were solving the right problem. When we first got our project, we discovered that the existing software out there only ran on windows, and we wanted to solve this problem, by making a software that is no restricted, and can run on the right computers. We also want it to have an option to implement a framework that can grow.The team is considering adding new features in the future, making it available on mobile. We are still considering different options, and no decision is set in stone yet. Licensing the product is also a possibility, in the future as for selling this product and filling the market gap left by the previous tool.

2.0 Report Revision History

2.1 Changes in Version 1.0

Table of Contents
1.0 Abstract
1.1 Project Goals
3.2 Needs(why)
3.3 Objectives(what)
4.1.4 Non-Functional Requirements
4.2 System Requirements
4.2.1 Functional Requirements
4.2.2 Non-Functional Requirements
5.1 Relevant Development Frameworks
5.2 Relevant Solution Techniques
12.1 Glossary of Relevant Domain Terminology
13.0 Reference Documents

2.1.1 Changes in Version 1.5

Table of Contents
5.3 Broader Impacts-pending
6.0 System Design-updated
6.2 Structural Design-pending
6.4 Behavioral Design-updated
9.0 Challenges & Open Issues

TE System Testing(Test Case Design)-updated

2.1.2 Changes in Version 2.0

Table of Contents
5.3 Broader Impacts
6.0 System Design
6.1 Architectural Design
6.2 Structural Design
6.3 User-Interface Design
6.4 Behavorial Design
6.5 Design Alternatives & Design Rationale
7.4 Justification for Chosen Approaches
9.0 Challenges & Open Issues
TE System Testing (Test Case Design)

2.1.3 Changes in Version 2.5

Table of Contents
1.2 Potential Impacts
Page Numbers
3.1 Business Background
3.3 Objectives
4.1.4. Non-Functional Requirements

2.14 Changes in Version 3.0

Table of Contents
2.0 Report History (update) (edited)
6.0 Quality of system modeling (UML System Models)
6.2 Structural Design
6.3 User-interface Design
9.0 Challenges update
10.0 System Manuals

Appendix T System Implementation (Code correlates with system design)
Appendix TE System Testing

2.15 Changes in Version 3.5

Table of Contents
2.0 Report History
6.0 System Design
6.1 Architectural Design
6.3 User-Interface Design
12.0 References
Appendix T System Implementation
Appendix TE System Testing

2.16 Changes in Version 4.0

Tables of Contents
2.0 Report History
6.0 System Design
6.1 Architectural Design
6.4 Behavorial Design
7.5 System Implementation
8.0 Testing
10.0 System Manuals

2.17 Changes in Version 5.0

Tables of Contents
3.4 Lessons Learned
4.1.3.1 Project Scope - Activity Diagram
6.4 Behavioral Design – Activity Diagram
4.4 Key Challenges (Requirements)
4.4.1 Compatibility Across Operating Systems
4.4.2 Optimizing Performance for Large Data Sets
4.4.3 Integration with External Tools
4.4.4 Ensuring Usability
6.6 Key Challenges (System Design)
6.6.1 Balancing Flexibility and Complexity
6.6.2 Achieving Platform Independence
6.6.3 Managing Interactions and Dependencies

7.6 Key Challenges (System Implementation)
7.6.1 Intergation of Libraries and Technologies
7.6.2 Ensuring Code Quality and Maintainability
7.6.3 Managing Code Versions and Dependencies
7.6.4 testing and Debugging
8.0 Key Challenges (Testing)
9.0 Challenges & Open Issues
11.0 System Testing

3.0 Problem Statement

3.1 Background Information:

(TARGEST) technical abstraction report generator extraction software tool is specialized in reading word documents with the Python/Docx library, generating excel reports with the Python/xlwings library, and displaying diagrams flow of requirements from documents A to B to C, etc. The goal of TARGEST is to make a graphical representation of the interaction of documents at their respective level. The process of this software is to generate multiple reports such as list of all tags used, lists of all tags that have no children, lists of tags that have no parents, lists of tags that are not tested, lists of duplicate tags, and text reports showing parents adjacent to children, to provide a means to check the parent and child tags are correctly tagged. The software will be able to have these reports in Excel by creating them using xlwings, Python based libraries that makes it easy to call Python from Excel. It addition they help create reading and writing to and from Excel using Python very easily.

3.2 Needs(why):

Tandem Diabetes Care works with the FDA(Food Drug Administration) to confirm the use of their medical software in devices like insulin pumps. Needing confirmation from the FDA, the medical software needed to be developed in detail. By developing this software in detail and with patience, Tandem was able to have a successful process of confirmation. In order for requirement tracing to be a successful process we need to manage the requirements through documentation based on the product's development process.

With our software, we are able to have key phrases known as tags that work to organize these requirements. The reason we need to tag documents is in order to create a structured environment

where the tags will help users see the distinction and see the connection between the documents. Grouping these tagged documents in their respective order will allow us to control and monitor the lifecycle of requirements. This will lead to consistency between the requirements in each document. The need for grouping documents by tags is to allow a company to have a clear layout of the user or the system requirements. Tracing will lead to better testing and allow users to identify defects or successes occurring in this lifecycle of requirements.

3.3 Objectives(what):

(TARGEST) technical abstraction report generator extraction software tool shall work on both mac and on PC. The software needs to be available A% of the time and we will use proper mechanism to test this requirement. It should ideally not have any bugs and shall have a good GUI. The software shall be able to read the Word documents and reconcile the tags and requirements. It shall be able to generate multiple reports. To read the word documents, it shall use Python/Docx library, and the excel report will be generated with Python/xlwings. Graphical representations of the interaction of the documents will be created at the highest level; These diagrams will show the flow of requirements and each level will have more than one document. TARGEST will aid Tandem in the process of document requirements management. We will be saving Tandem a week worth of work which can save the company lots of money.

3.4 Lesson Learned:

Jan:

Agile Development and Iterative Process:

- Embracing an agile mindset and adapting to change
- Regularly reviewing and refining project scope and requirements

Focus on User experience and Usability:

- Researching target audience needs and preferences.
- Refining of the user interface

Learning New technologies and Tool

- Adapting to new tools as needed
- Leveraging online resources, tutorials to learn quickly.

Stephania:

- Learning Python and Py Unit Testing-for writing more effective code and testing it to ensure that software is reliable
- Having a better grasp and understanding of Agile development- by breaking our project down into smaller, manageable tasks and regularly checking in on team's progress
- Incorporating project management into our project- to minimize the risk of delays, miscommunications, or misunderstandings and ultimately improving the quality and efficiency of our software development project.

Adrian:

Time Management:

- Having tasks for every sprint helped us manage our time better, especially if there was a deadline for priority tasks.
- Small release cycles helped us build momentum that benefited us in terms of hours of needed time spent working on features was at a minimum towards the final sprints.

Platforms:

- Learned to be mindful to needing to implement alternatives since testing and prototyping methods were being done on both MacOS and Windows.

4.0 Requirements

4.1 User Requirement

User Requirements are validated through black-box testing which occurs during the transformation portion of our problem statements. These requirements are organized by the end user. The user will provide the necessary files and the software will generate reports.

4.1.2 User Groups

The main audience of this document includes users utilizing either Mac IOS or Windows OS.

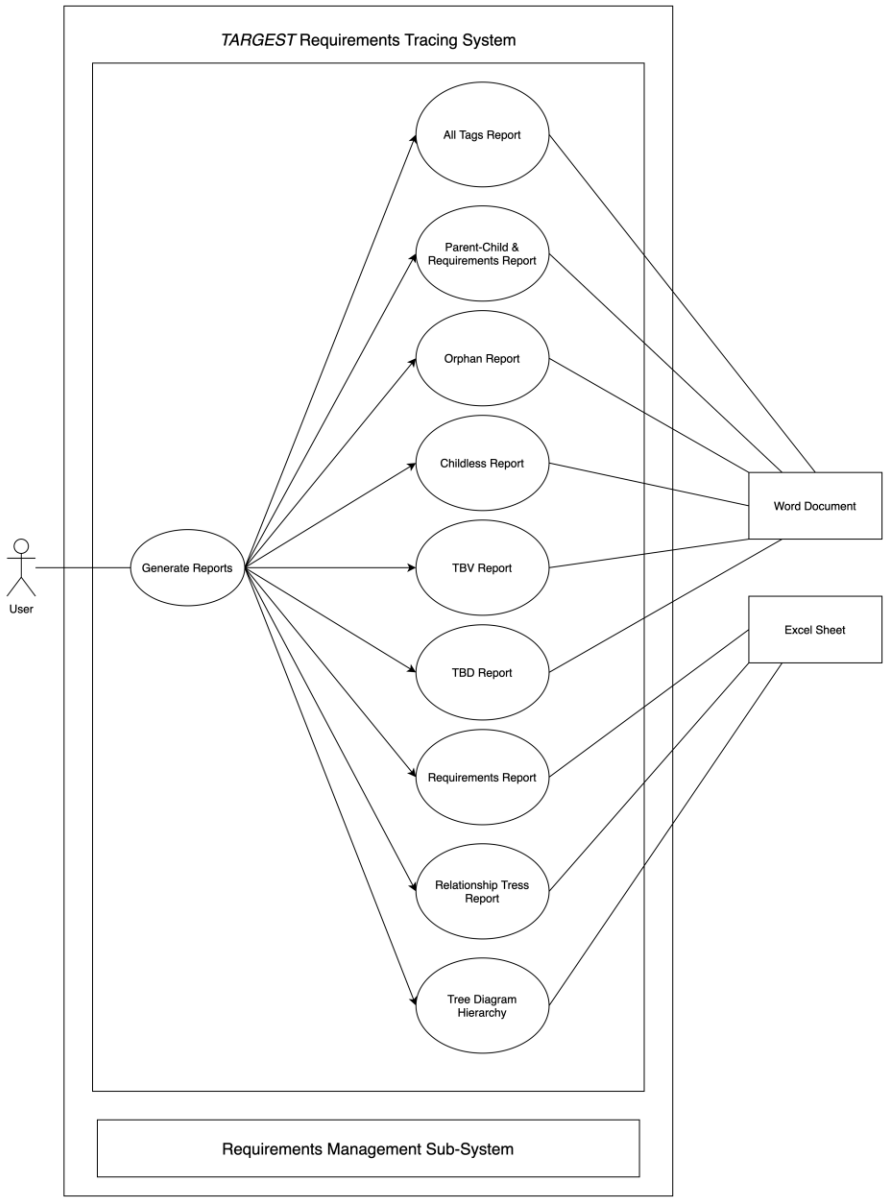
Key Users: Developers are supposed to use this document in the development phase to the structure and design of each component.

4.1.3 Functional Requirements (User)

The user shall be able to experience an instantaneous interface with fast runtime regardless of the amount of documentation generated. This is something that a user needs to perform their work.

4.1.3.1 Project Scope

The user will give a list of documents to the software, *TARGEST* will then read the files, and the software shall then generate the reports. The motivation for this diagram is to illustrate how the software tool will be used to demonstrate parsing documents for tags and linking documents based on the tags and requirements hierarchy. Our tool will be able to match a tag with a given set of tags depending on what report is chosen to generate.


TARGEST Requirements Tracing System
All Tags Report
Parent-Child & Requirements Report
Orphan Report
Childless Report
TBV Report
TBD Report
Requirements Report
Relationship Tress Report
Tree Diagram Hierarchy
Generate Reports
User
Word Document
Excel Sheet
Requirements Management Sub-System

4.1.3.2 User Scenarios

This use case scenario is a path that leads to success for the objective of TARGEST.

Table 4.1. Use Case Index Table

Use Case ID: **UC-001**
Use Case Name: case 1
Level: Primary task
Version: 0.2

Use Case ID: **UC-002**
Use Case Name: Case 2
Level: Primary task
Version: 0.3

Use Case ID: **UC-003**
Use Case Name: Case 3
Level: Summary
Version: 0.2

Use Case ID: **UC-004**
Use Case Name: Case 4
Level: Summary
Version: 0.2

Use Case ID: **UC-005**
Use Case Name: Case 5
Level: Primary task
Version: 0.1

Use Case ID: **UC-006**
Use Case Name: Case 6
Level: Primary task
Version: 0.1

Use Case ID: **UC-007**
Use Case Name: Case 7
Level: Summary
Version: 0.1

4.1.3.3 User Functional Requirements

User Functional Requirements: UF-A
Table 4.9 The system shall allow the user to run the software on both OS and Windows.
Priority: High
User Functional Requirements: UF-B
Table 4.10 The user shall be able to experience an instantaneous interface with fast runtime regardless of the amount of documentation generated.
Priority: High
User Functional Requirements: UF-C
Table 4.11 The software should support the user generating multiple reports such as lists of all tags used.
Priority: High
User Functional Requirements: UF-D
Table 4.12 The software should support the user generating multiple reports such as lists of all tags that have no children
Priority: High
User Functional Requirements: UF-E
Table 4.13 The software should support the user generating multiple reports such as lists of tags that have no parents.
Priority: High
User Functional Requirements: UF-F
Table 4.14 The software should support the user generating multiple reports such as lists of tags that are not tested.
Priority: High
User Functional Requirements: UF-G
Table 4.15 The software should support the user generating multiple reports such as lists of duplicate tags.
Priority: High
User Functional Requirements: UF-H
Table 4.16 The software should support the user generating multiple reports such as text reports showing parents adjacent to children.
Priority: High

4.1.4 Non-Functional Requirements (User)

The software can still work even if these requirements are not met, but they will not meet the user expectations or the needs of the business.

4.1.4.1 Product: Usability Requirements

User Nonfunctional Requirements: UP-01

Table 4.17 The system should support cross platforms and should be able to be used on a computer device.
Priority: Highest

4.1.4.2 Organizational: Development Requirements

User Nonfunctional Requirements: UO-01

Table 4.18 The application shall consist of 5 documents, 5 embed tags, and 5 test cases.
Priority: High

4.2 System Requirements

System Requirements are created with white-box testing needed for verification. These test suites are established through certain design structures that are determined by certain quality attributes in respects to security, usability, and response time.

4.2.1 Functional Requirements

Functional requirements will meet the very technical requirements in respects to what the software is doing with its product services. These requirements are TARGEST features or functions that will be implemented to enable users to accomplish their tasks and describe system behavior.

4.2.1.1 System Functional Requirements

System Functional Requirements: SF-A-01

Table 4.19. The system shall be able to run on multiple platforms.
Priority: High

System Functional Requirements: SF-B-01

Table 4.20. The system shall have an optimum graphical user-interface.

Priority: High
System Functional Requirements: SF-B-02
Table 4.21. The system shall display the number of errors.
Priority: High
System Functional Requirements: SF-B-03
Table 4.22. The system shall display number of successes.
Priority: High
System Functional Requirements: SF-C-01
Table 4.23. The system shall generate a list of all tags using xlwings.
Priority: High
System Functional Requirements: SF-D-01
Table 4.24. The system shall generate a list of all tags that have no children.
Priority: High
System Functional Requirements: SF-E-01
Table 4.25. The system shall generate a list of tags that have no parents using docx or xlwings.
Priority: High
System Functional Requirements: SF-F-01
Table 4.26. The system shall generate a list of tags that are not tested using docx or xlwings.
Priority: High
System Functional Requirements: SF-G-01
Table 4.27. The system shall generate a lists of duplicate tags.
Priority: High
System Functional Requirements: SF-H-01
Table 4.28. The system shall generate a lists of texts showing parents adjacent to children.
Priority: High

4.2.2 Non-Functional Requirements

Through the development process, non-functional requirements are created and verified by three categories. These include Product to determine the quality of services, Organizational to establish the software's development structure, and External which is factored in by safety or security, cultural or social, or political as well.

To describe the system's operation capabilities and constraints in attempts to improve its functionality.

4.2.2.1 Product: Usability Requirements

System Non-Functional Requirements: SP-01-01
Table 4.29. The system shall run on both OS and Windows platforms
Priority: High

4.2.2.2 Organizational: Development Requirements

System Non-Functional Requirements: SO-01-01

Table 4.30. The system supports external tools such as Pydoc, xlwings, and PyUnit or ProUnit.
Priority: High

4.3 Requirements Trace Table

Table 4.31 Mapping from user requirements to system requirements
UF-A = SF-A-01
UF-B = SF-B-01, SF-B02, SF-B-03
UF-C = SF-C-01
UF-D = SF-D-01
UF-E = SF-E-01
UF-F = SF-F-01
UF-G = SF-F-01
UF-H = SF-H-01
UO-01 = SO-01-01
UP-01 = SP-01-01

4.4 Key Challenges (Requirements)

This sub- section documents the challenges encountered during the development process of the TARGEST software and the efforts made to address these challenges.

4.4.1 Compatibility Across Operating Systems

Challenge: Ensuring compatibility and seamless performance of the software on both Mac OS and Windows OS platforms.
Effort: We utilized cross-platform programming languages and libraries, such as Python, to guarantee that the software runs smoothly on both operating systems. We also conducted extensive testing on both platforms to identify and fix any issues.

4.4.2 Optimizing Performance for Large Data Sets

Challenge: Providing an instantaneous interface with fast runtime regardless of the amount of documentation generated.

Effort: We applied optimization techniques, such as code refactoring, parallel processing, and efficient data structures, to improve the software's performance. We also conducted rigorous stress testing to identify potential bottlenecks in the system and address them accordingly.

4.4.3 Integration with External Tools

Challenge: Ensuring seamless integration with external tools like Pydoc, xlwings, and PyUnit or ProUnit, as specified in the requirements.
Effort: We collaborated closely with the developers of these external tools to understand the necessary protocols and best practices for integration.

4.4.4 Ensuring Usability

Challenge: Creating an intuitive and user-friendly graphical user interface (GUI) that meets the needs of a diverse user base.
Effort: We conducted thorough user research to understand the preferences and requirements of the target user groups.

5.0 Exploratory Studies

5.1 Relevant Development Frameworks

Flask is framework we are considering implementing in the future. Flask is a lightweight Python web framework that provides useful tools and features for creating web applications using Python, if we decide to make a web application for our software in the future. Framework decisions not yet finalized.
TKinter is another python framework we are considering implementing in our software in the future. This framework can provide us with a simple way of creating GUI elements using widgets found in TK toolkit. These widgets would then be used to construct buttons and menu fields for our application, so that the user can potentially in the features have the options to choose which word documents will be used by application when generating the reports.

5.2 Relevant Solution Techniques

Xlwings is an open source package that allows you to automate Excel with Python on Windows and macOS. This solution technique is favorable when working with python and excel together by synchronously being able to exchange data.

Python-Docx is a Python library for creating and updating Microsoft Word (.docx) files. Python-docx library has functions that allow us to implement techniques to illustrate and organize requirements. It also works well with xlwings.
Regex(Regular Expressions) allows us to apply techniques to analyze for patterns in text strings. Regex is a powerful solution tool that is able to define our own search criteria for specific patterns in documents that fit our product's needs.
Pandas is a software library written for python. Pandas is used for data manipulation and analysis.
Matplotlib is a comprehensive library for creating static, animated and interactive visualizations.

5.3 Broader Impacts:

Local Impact

The TARGEST software is designed to benefit local companies with the sole purpose of conducting requirements traceability. Our software will provide local companies with the following advantages: ensure that their organizations do not waste time and resources in repeating tasks, compiling with established industry standards and most importantly ensuring that final deliverables are directly tied to their company's initial business needs.

Global Impact

Prior to TARGEST, writing software for real world applications in managing requirements have hold a very critical role in a sense that they go hand in hand when undergoing system consideration. As a result of this, a manual approach was applied for the purpose of traceability and in handling requirements management via software. However as the requirements evolved and tasks became more complex, this method proved to be time consuming, very inefficient and outdated. Targest came about to meet that need and fill in for the lack of software available that perform the functionality of processing and file generation for specific tasks that do not require to be manually done. What was already available wasn't as accessible in generating documents efficiently due to problems related to a bug causing the software to be prone to crashing. It was a need for this kind of software, so we came with a solution for that. The purpose and goal of this project is to minimize workload, time management, task handling and ensure a user friendly experience. We have teamed up with Tandem Diabetes, a local diabetes care company that offers products and services to the diabetic community. Through mutual collaborations, TARGEST have designed this product to ensure productivity and an efficient way of handling sensitive patients records and data, and to provide the best experience for the user and the customer.

Societal Impact

The TARGEST software is very helpful in many ways since it will provide context for development teams to create reliable, safe and approved products at any company for commercial

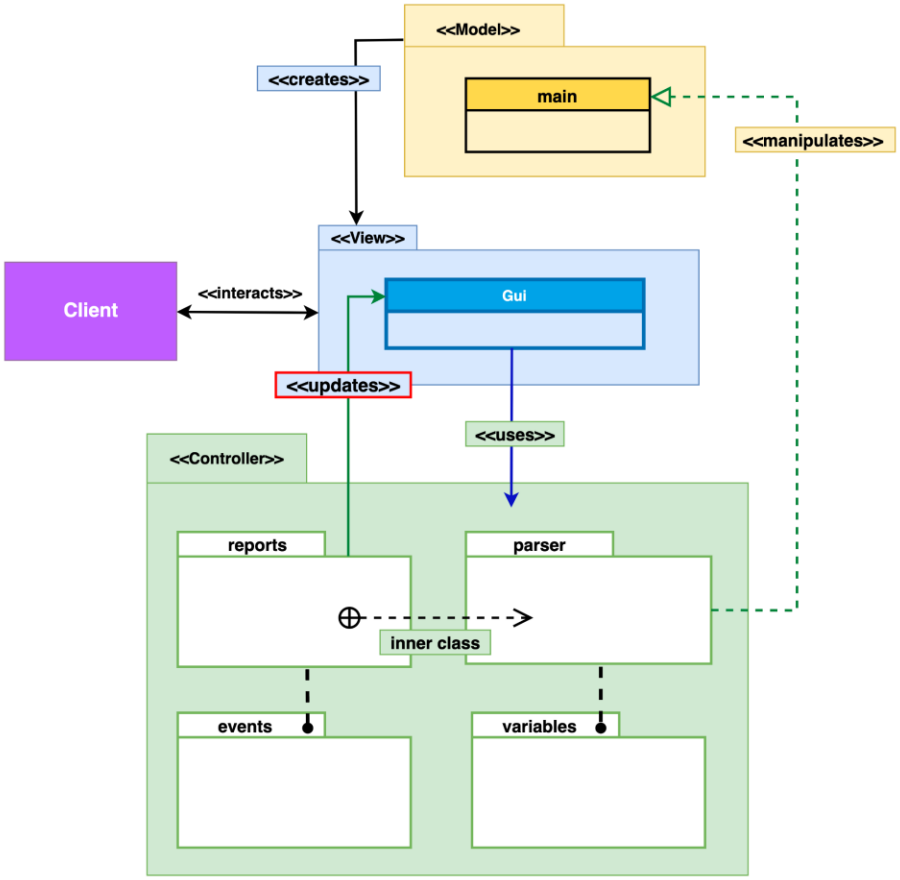
or non-commercial purposes. We have designed our application to define the expectations to any testing team in terms of implementation and specification. This will guarantee companies that there not only building the product right, but are also building the right product. TARGEST will help put to perspective the size, complexity, and risks of any project, while still ensuring ensure compliance with industry standards, functional safety, and helping to minimize the risk of negative outcomes and maximize productivity.

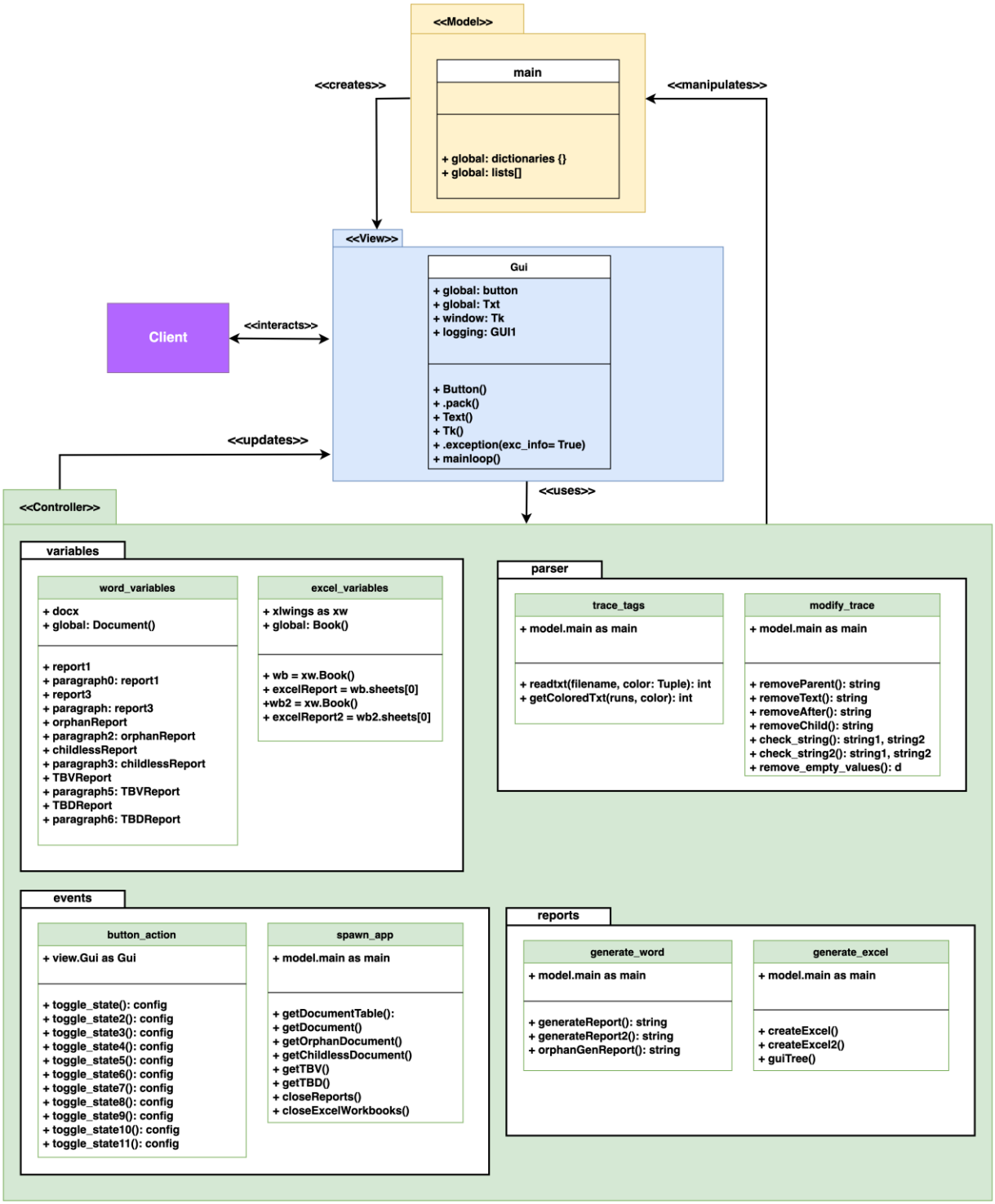
6.0 System Design

6.1 Architectural Design

Model-View-Controller (Level 2 Design)

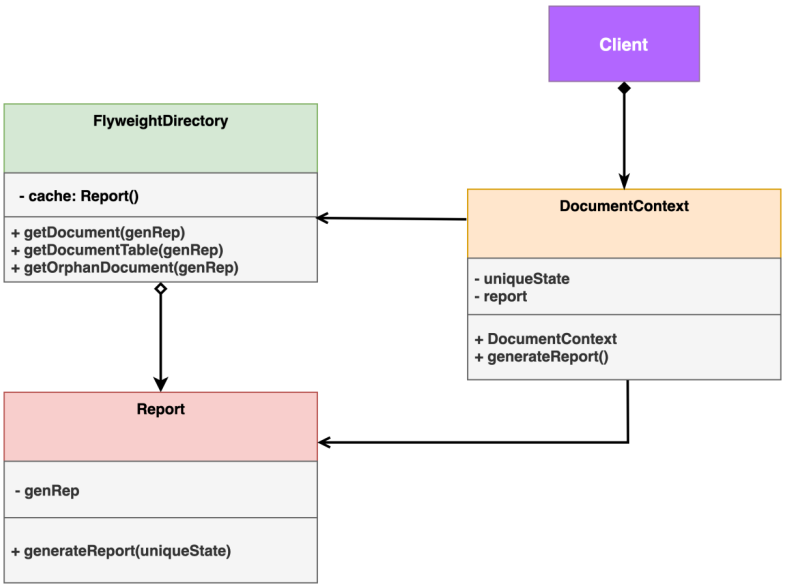
The overall architecture of TARGEST follows a Model-View-Controller (MVC) pattern. This design method promotes loose coupling by separating the application into three interconnected components. Our Model represents the data and business logic of the application. The view displays data to the user and handles user interaction. In our Controller component, we have folders that serve a specific purpose for managing the flow of data between the Model and the View. The variables folder contains classes for managing variables in different formats such as Word and Excel. The parser folder contains classes for parsing trace tags and modifying trace information. The events folder contains classes for handling button actions and new instances of the application. The reports folder contains classes for generating reports in Word and Excel formats through iteration.


<<Model>>
<<creates>>
main
<<manipulates>>
<<View>>
Client
<<interacts>>
Gui
<<updates>>
<<uses>>
<<Controller>>
reports
parser
inner class
events
variables


<<Model>>
main
+ global: dictionaries {}
+ global: lists[]
<<creates>>
<<manipulates>>
<<View>>
Gui
+ global: button
+ global: Txt
+ window: Tk
+ logging: GUI1
+ Button()
+ .pack()
+ Text()
+ Tk()
+ .exception(exc_info= True)
+ mainloop()
Client
<<interacts>>
<<updates>>
<<uses>>
<<Controller>>
variables
word_variables
+ docx
+ global: Document()
+ report1
+ paragraph0: report1
+ report3
+ paragraph: report3
+ orphanReport
+ paragraph2: orphanReport
+ childlessReport
+ paragraph3: childlessReport
+ TBVReport
+ paragraph5: TBVReport
+ TBDReport
+ paragraph6: TBDReport
excel_variables
+ xlwings as xw
+ global: Book()
+ wb = xw.Book()
+ excelReport = wb.sheets[0]
+wb2 = xw.Book()
+ excelReport2 = wb2.sheets[0]
parser
trace_tags
+ model.main as main
+ readtxt(filename, color: Tuple): int
+ getColoredTxt(runs, color): int
modify_trace
+ model.main as main
+ removeParent(): string
+ removeText(): string
+ removeAfter(): string
+ removeChild(): string
+ check_string(): string1, string2
+ check_string2(): string1, string2
+ remove_empty_values(): d
events
button_action
+ view.Gui as Gui
+ toggle_state(): config
+ toggle_state2(): config
+ toggle_state3(): config
+ toggle_state4(): config
+ toggle_state5(): config
+ toggle_state6(): config
+ toggle_state7(): config
+ toggle_state8(): config
+ toggle_state9(): config
+ toggle_state10(): config
+ toggle_state11(): config
spawn_app
+ model.main as main
+ getDocumentTable():
+ getDocument()
+ getOrphanDocument()
+ getChildlessDocument()
+ getTBV()
+ getTBD()
+ closeReports()
+ closeExcelWorkbooks()
reports
generate_word
+ model.main as main
+ generateReport(): string
+ generateReport2(): string
+ orphanGenReport(): string
generate_excel
+ model.main as main
+ createExcel()
+ createExcel2()
+ guiTree()

6.2 Structural Design
Flyweight Pattern

- With the Flyweight Pattern were able to control a large number of similar objects within an application, which improves performance, reduces the memory footprint, and saves RAM. Since we are dealing with documents and tags within those documents, a flyweight pattern can help with object caching in a repeatingState private method without interfering with different functionalities depending on the features designed.

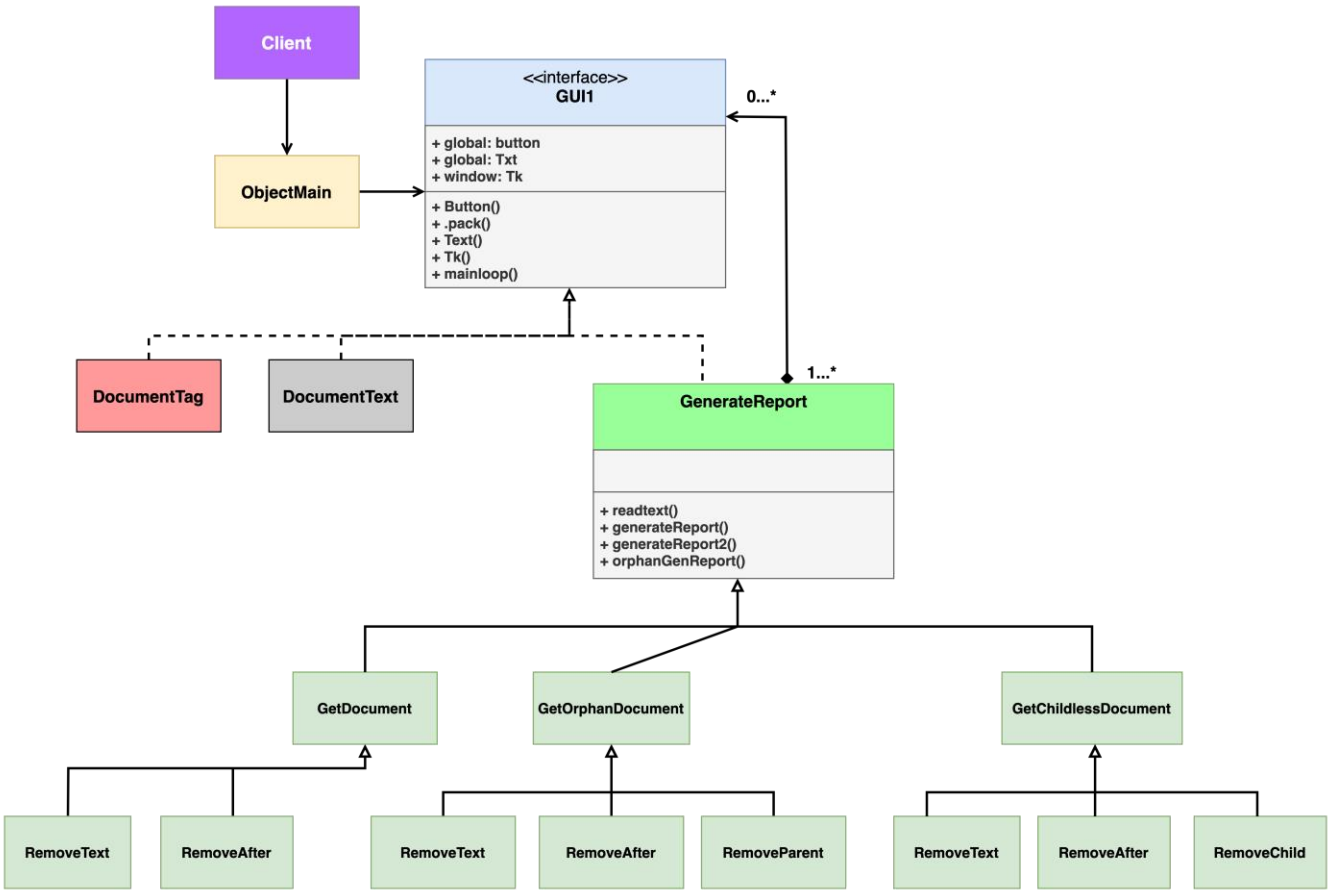


Design Patterns	**Model-View-Controller**	*Flyweight*	Visitor	Singleton	Strategy
Comp	Separates an application into three interconnected components: the Model, the View, and the Controller.	Shares objects to reduce memory usage by reusing common data between multiple objects.	Allows adding new operations to an existing object structure without modifying the objects themselves.	Ensures that a class has only one instance and provides a global point of access to it.	Defines a family of interchangeable algorithms and encapsulates each one, making them interchangeable at runtime.
Chosen or not Chosen	Chosen over other design patterns for its clear separation of concerns, making it easier to maintain and modify different aspects of an application independently.	Not chosen over MVC because it focuses on optimizing memory usage, whereas MVC provides a comprehensive architectural framework for developing complex applications.	Not chosen over MVC because it focuses on optimizing memory usage, whereas MVC provides a comprehensive architectural framework for developing complex applications.	Not chosen over MVC because Singleton addresses the instantiation of a single object, while MVC focuses on the overall structure and organization of an application.	Not chosen over MVC because Strategy focuses on algorithmic variations, whereas MVC focuses on the overall architectural structure and separation of concerns in an application.

6.3 User-Interface Design

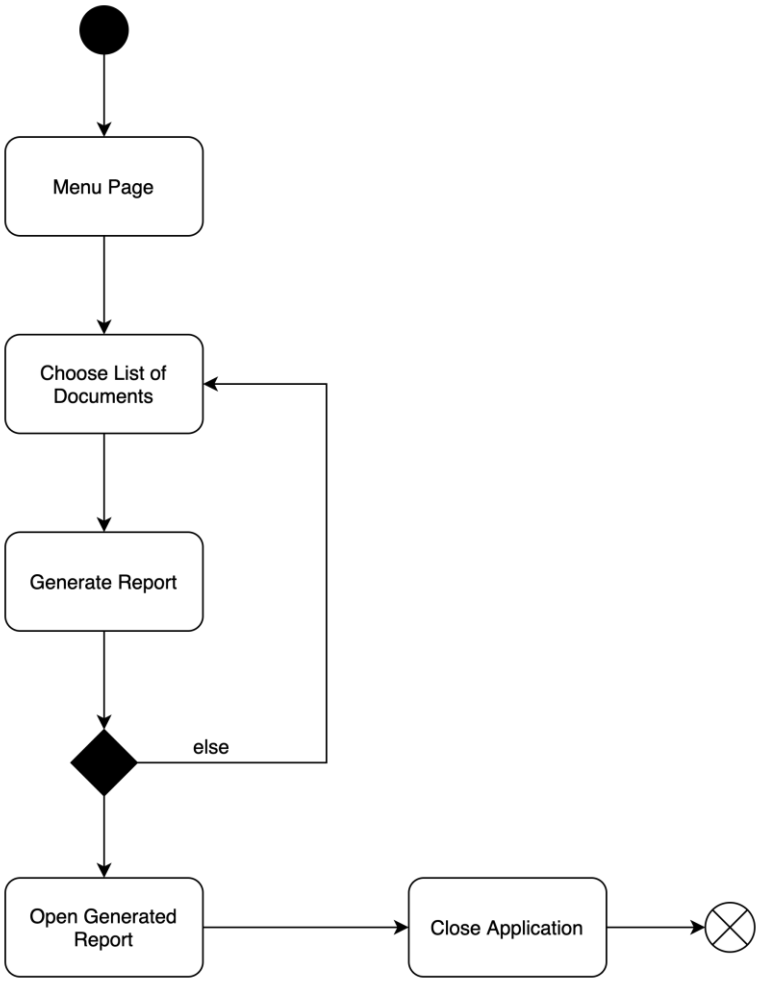
Visitor Pattern

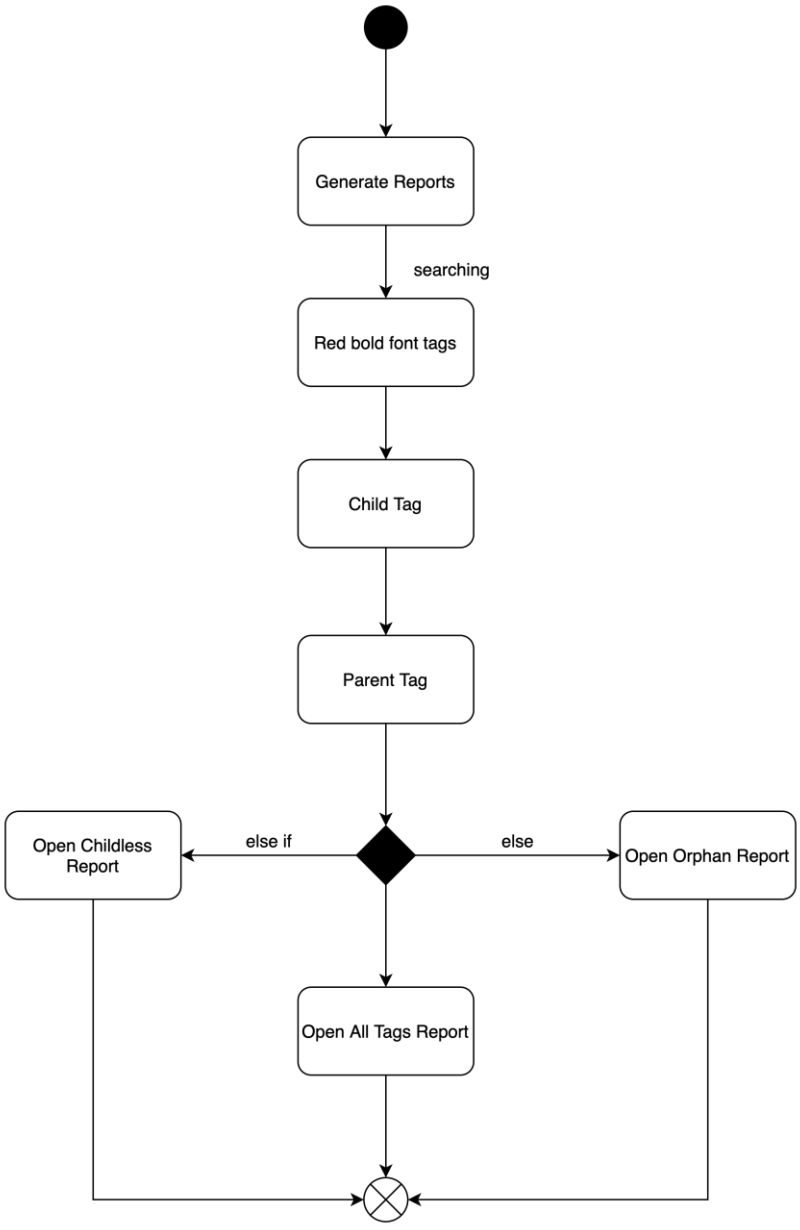
- The Visitor Pattern allows for defining new operations under the GenerateReport class without needing to modify the ObjectMain class which is considered the visitor interface. This pattern implementation allows the Client to use existing item classes that aren't affected by updates and modifications.

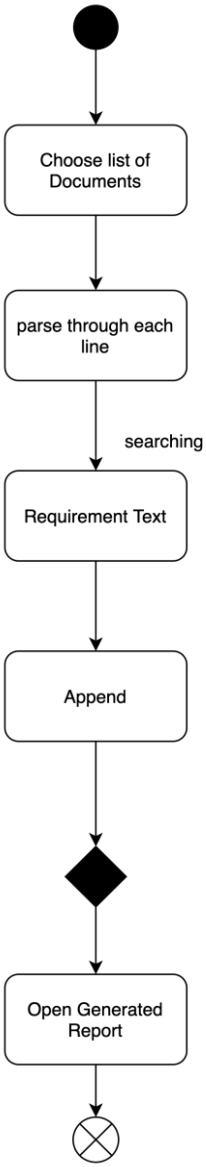


6.4 Behavioral Design
Activity Diagram

Our behavioral design captures the system dynamics as a activity diagram. In our activity diagram, it explains the current state of our program and how it works as a user. In the initial state the user starts off in the menu page where they are able to choose a document. Once the ‘Choose Document’ action node button is chosen, we then proceed to choose the next action node of ‘Generate Report’ button which involves generating a report from the chosen document. As a user, we then are given a control: decision to either choose another document or open the current document that was generated. If the user chooses to open generated report, then they are able to then close the application.

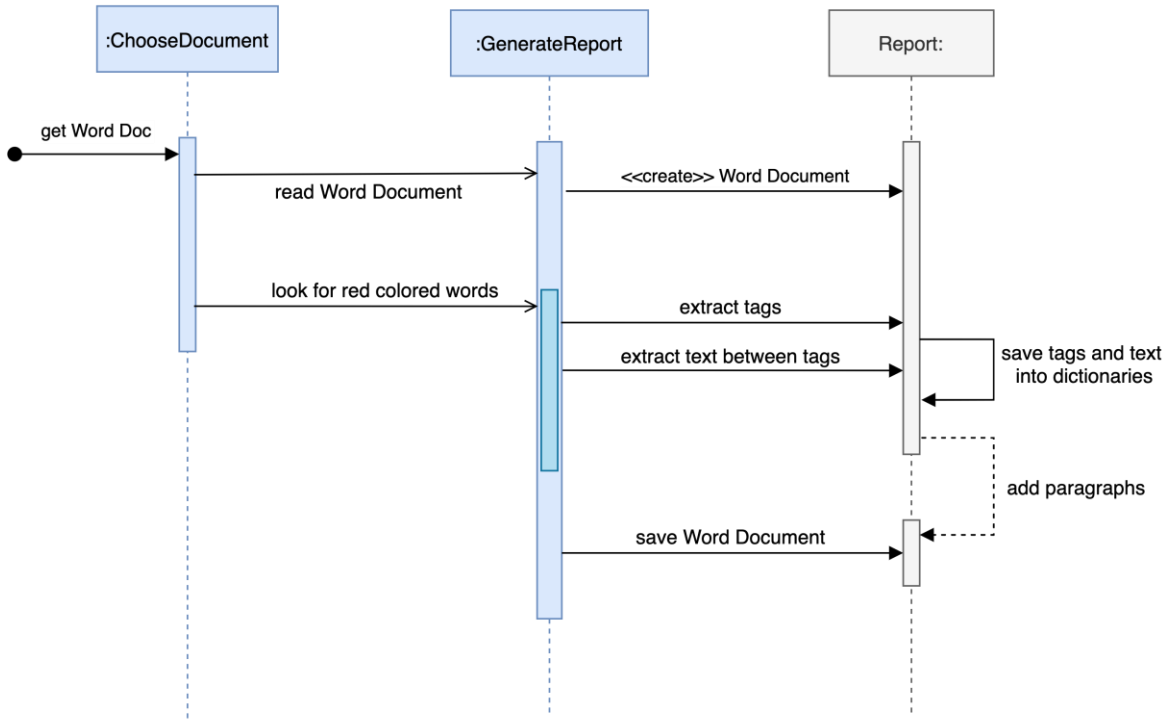


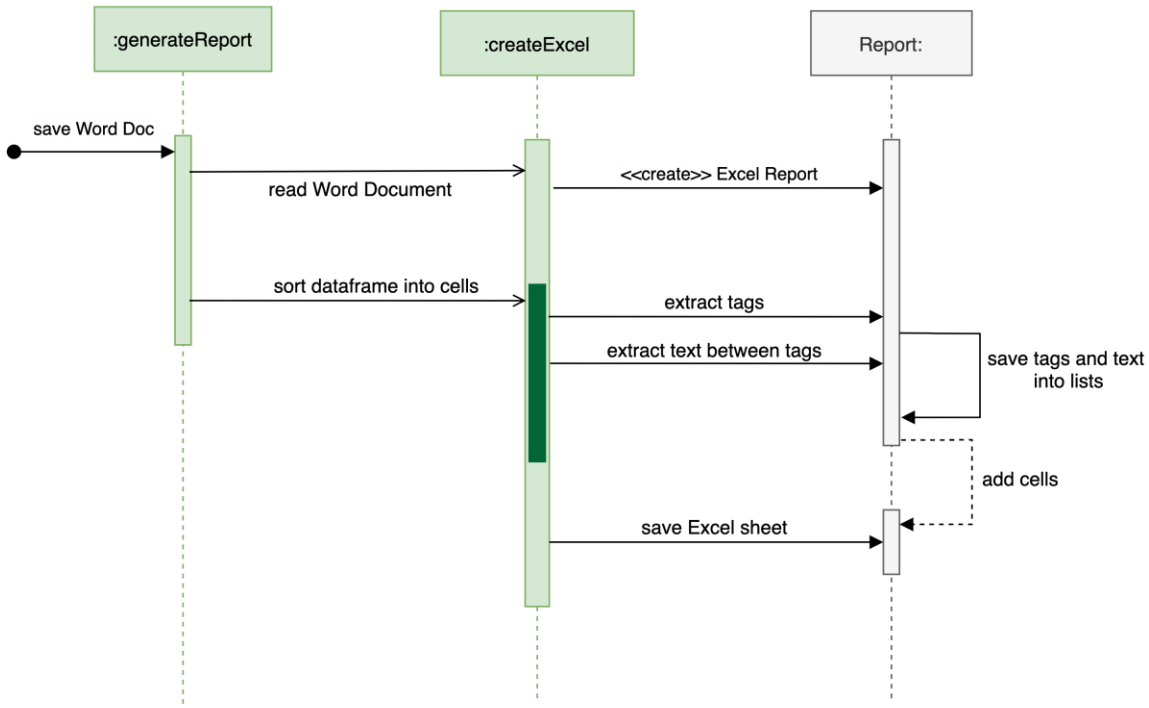

Generate Reports
searching
Red bold font tags
Child Tag
Parent Tag
Open Childless Report
else if
else
Open Orphan Report
Open All Tags Report


Choose list of Documents
parse through each line
searching
Requirement Text
Append
Open Generated Report

Sequence Diagram

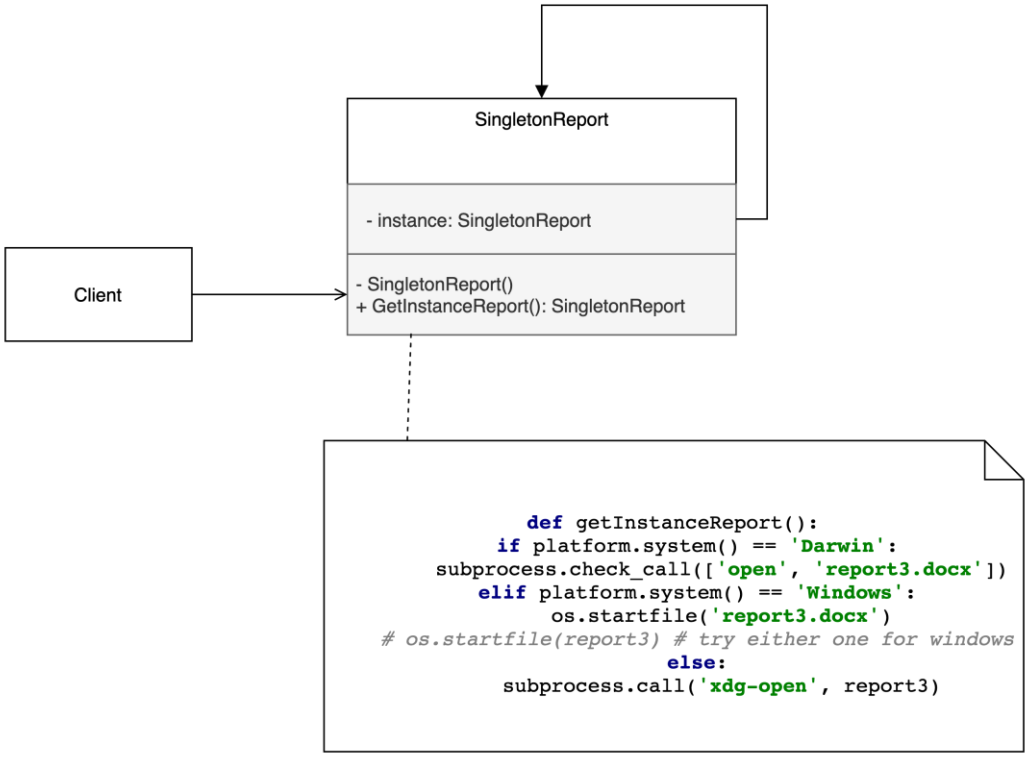
With our sequence diagram, we are able to illustrate the certain tasks that are taking place when generating a report. Here with our sequence diagram, we approach this at a low-level design to contribute to displaying the use of interaction modeling.





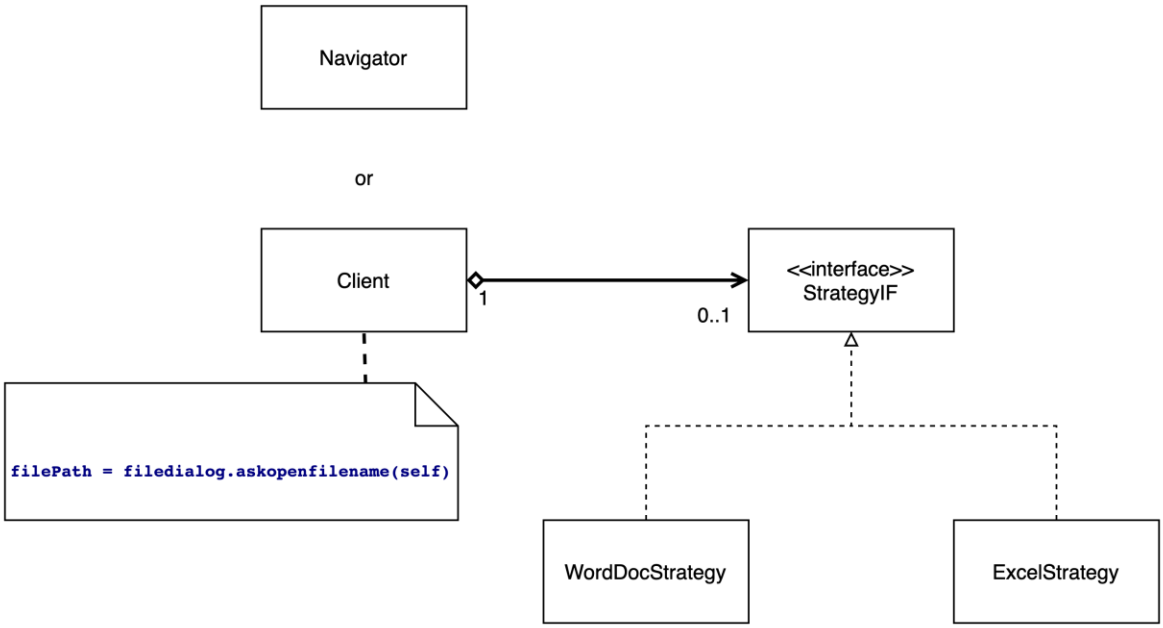
6.5 Design Alternatives & Design Rationale

A design alternative that we were considering was the Singleton Design Pattern because of its philosophy of only using one instance to execute objects exclusively. Targest is not tied to one OS. The motivation for this diagram was to give a better solution to implementing a version that is able to run on both Windows and Mac-OS since previous versions only ran on Windows and a had a poorly designed interface. This will allow the tool to have usability expansion.



Another design alternative was the strategy pattern. The client decides the actual implementation used at runtime. This pattern makes it easier to extend. And also prevents the conditional

statements.(if, else...). You also end up with a cleaner code, because you separate the concerns into classes, Word Doc strategy, Excel strategy etc. (a class to each strategy).



6.6 Key Challenges (System Design)

This sub-section documents the challenges encountered during the system design process of the TARGEST software and the efforts made to address these challenges.

6.6.1 Balancing Flexibility and Complexity

Challenge: Creating a flexible system design that can easily accommodate future modifications and updates while avoiding unnecessary complexity.

Effort: We carefully evaluated various design patterns and approaches to strike the right balance between flexibility and complexity, ultimately opting for the MVC design pattern.

6.6.2 Achieving Platform Independence

Challenge: Ensuring that the TARGEST software runs seamlessly on Windows and Mac-OS. Effort:
We considered various design alternatives, including Singleton and Strategy patterns, to enable platform independence and ultimately chose MVC as our main design pattern.

6.6.3 Managing Interactions and Dependencies

Challenge: Effectively managing interactions and dependencies among various components of the system to ensure smooth operation and maintainability.

Effort: We used behavioral and structural design approaches, such as activity and sequence diagrams, to model system dynamics and interactions, enabling a clear understanding of component relationships and dependencies.

7.0 System Implementation

7.1 Programming Languages and Tools

Language:

- Python3

Tools:

- Xlwings
- Python-docx
- Tkinter
- Pandas
- Matplotlib

7.2 Coding Conventions

Naming variables by separating words with a single capitalized letter, and the first word starting with lowercase, following by next word starting with a single capitalized letter.

7.3 Code Version Control

Python 3.9 version

Xlwings 0.29.0 version
Python-docx 0.8.11 version
Pandas 1.3.5 version
Matplotlib 3.5.3 version

7.4 Justification for Chosen Approaches

The chosen approaches we decided to take based upon the design alternatives and design rationale was implementing the Facade pattern, Flyweight pattern, and Visitor pattern. These 3 approaches deal with 2 structural patterns (Facade, Flyweight), and 1 behavioral pattern (Visitor). The structural patterns enable us to explain how to create objects and classes into larger structures while remaining open to adding new modifications. With the behavioral pattern this allows us to be more concerned with what algorithms are being used and the role each object is assigned to. In our case, we deal with finding tags in existing documents and generate a specific report that will organize it depending on the test case our program is given.

7.5 System Implementation

TARGEST is designed to streamline the process of creating and managing requirements documents, facilitating better communication and understanding among engineers, and ensuring accurate documentation and test coverage. We will cover the technologies and libraries used, as well as the key components of our application.

The development of TARGEST followed an iterative approach, with each component being developed and tested independently before being integrated into the main application. Our algorithm approaches are using basic file I/O(Input/Output) operations and string manipulation methods to handle files and process text data. The implementation process involved:

a. GUI prototyping
b. Requirements gathering and analysis
c. Design of application architecture and components
d. Selection of appropriate libraries and technologies
e. Implementation of individual components
f. Integration of components into the main application
g. Testing and debugging
h. Finished Development of the GUI
i. Final testing and validation

In this section, we have outlined the implementation process for TARGEST, detailing key components that make up our application. By leveraging Python and a range of powerful libraries, we have successfully developed a tool that streamlines requirements document management and ensures clear communication among engineers, leading to smoother and more efficient process.

7.6 Key Challenges (System Implementation)

This sub-section documents the challenges encountered during the system implementation process of the TARGEST software and the efforts made to address these challenges.

7.6.1 Integration of Libraries and Technologies

Challenge: Integrating various libraries and technologies, such as Xlwings, Python-docx, Tkinter, Pandas, and Matplotlib, to create a cohesive and functional application.
Effort: We conducted thorough research on the chosen libraries and technologies to understand their compatibility and the best practices for integration.

7.6.2 Ensuring Code Quality and Maintainability

Challenge: Adhering to coding conventions and best practices to ensure that the code is easily readable, maintainable, and scalable.
Effort: We established and followed a set of coding conventions, such as consistent naming conventions for variables and functions. We also conducted regular code reviews to ensure that the code adhered to these conventions and to identify areas for improvement.

7.6.3 Managing Code Versions and Dependencies

Challenge: Managing code versionsto ensure that the application remains stable and functional as libraries and technologies are updated.
Effort: We used version control tools and maintained a clear record of the specific versions of libraries and technologies used in the project. We made adjustments to the application as necessary to maintain compatibility.

7.6.4 Testing and Debugging

Challenge: Thoroughly testing and debugging the application to ensure that it meets all functional and non-functional requirements and operates without errors.
Effort: We employed a combination of unit testing, integration testing, and system testing to identify and fix any issues with the application.

8.0 Testing

Test strategy and preliminary test case specification are presented in this section.

8.1 Types of tests

Software will read the documents all (at once) and generate the reports based on specific tasks that the user chooses.

In order for the following document/report generation to take place, the user must first open up the application on their screen. Once the application is prompted, the user will click on the **"Choose list of Document"** where a new screen pops up and allows the user to select the document of their choice. Once document has been chosen, the user selects the **"Generate Reports"** button where the reports are generated.

Report Docx generation:

All Tags Table Report- Shows all the child tags and parent tags in each of the requrement documets, in a table forrmat. User selects the **"Open All Tags Table Report"** button to view this report on the application.

Child And Parent Tags Report- This report, lists out all the parent tag, and their child tags. User selects the **"Open Child and Parent Tags Report"** button to view this report on the application.

Oprhan Tags Report- For this report, all orphan tags are listed. User selects the **"Open Orphan Tags Report"** button to view this report on the application.

Childless Tags Report- For this report, all childless tags are listed. User selects the **"Open Childless Tags Report"** button to view this report on the application.

TBV Report- For this report, it shows all the pending tags that need to be verified whether their parents, child, or orphan. These tags have TBV in them. User selects the **"Open TBV Word Report"** button to view this report on the application.

TBD Report- For this report, all tags that needs to be done will be verified, these tags have TBD in them. User selects the **"Open TBD Word Report"** button to view this report on the application.

Excel report generation:

Requirements Excel Report:

In this report, all tags (leading and trailing tags), output whether it is a pass or fail , and their description are displayed in full view for the user to see. User selects the **"Open Requirements Excel Report"** button to view this report on the application.

Relationships Trees Excel Report:
This report is displaying all the different relationships, all the way from the orphan tags till the childless tags This shows all the generations in the different "families" in an excel format. User selects the **"Open Relationship Trees Excel Report"** button to view this report on the application.

Tree diagram generation:
Family Tree Diagran is created in a text view in the Gui. The different relationships, all the way from the orphan tags till the childless tags is displayed in the the scrollbar textview with lines between each tag displayed, to show which one connects to which one. This diagrams are similar to the Relationships Trees Excel Report, but had lines between thee tags to clearly show the conecctions. User selects the **"Create Family Tree"** button to view this report on the application.

An example success success scenario to open a specific report:

1. Run The application.
2. Click the "Choose a list of document" button on the gui, and Choose a list of documents with paths to the requirement documents.
3. Click the "Generate Reports" button on the Gui.
4. Choose the specific word report, excel report or family tree that the user wants to see:
   A. All Tags Table Report
   B. Child and Parent Tags Report
   C. Orphan Tags Report
   D Childless tags Report
   E. TBV Word Report
   F. TBD Word report
   G. Requirement Excel Report
   H. Relationship Trees Excel Report
   I. (Gui) Family Trees

8.2 Test Cases

- List of all tags used
- List of all tags that have no children
- List of tags that have no parents
- Lists of tags that are not tested
- Lists of duplicate tags

- Lists of TBD tags
- Lists of TBV tags
- Number of errors
- Number of successes
- New tbv tags added
- New tbd tags added
- New relations between tags
- New family trees
- New generations added

So far the majority of testing has been taking place by the running of the software in displaying certain tasks and adding tasks that generate the request reports that the user chooses. We were able to detect bugs and flaws through the process, when in the report generation we would see certain tags missing and not picked up by the software, misplaces of tags, lines between tags in the gui diagram trees missing, tags linking to wrong tags, Etc.

Currently, the team has been conducting testing through running the application and generating reports to identify issues which has enabled us to identify both the strengths and weaknesses of the application through report and document generation. However, we recognized the need for a more comprehensive testing approach to handle bugs and errors more effectively. Consequently, we have decided to implement Py Unit testing to achieve this goal. The test cases that we have created are specifically designed to target individual functions to ensure that they perform as expected and yield the desired results. Using Pyunit testing enhanced the overall quality and reliability of the software.

8.3 Key Challenges (Testing)

During the testing phase, we faced several challenges that required us to take a step back and reassess our approach. First, we performed a comprehensive code review as a team, which helped us identify areas that needed improvement. After determining what to keep and what to discard, we refactored the core code, focusing on the most critical components. This allowed us to streamline the testing process and avoid potential delays and bugs.

However, creating test cases proved to be a time-consuming task since we had to ensure that each test accurately simulated the application's functionalities and produced expected outputs. Although it took longer than expected, we recognized the importance of conducting thorough testing to ensure the application's quality and reliability.

9 Challenges & Open Issues

9.1 Challenges faced in in requirements engineering

1. Availability of Industry Mentor

It was a struggle for our team to find a designated time that worked for all of us to meet our faculty advisor. Eventually we were able to find a time and day that worked for all respective team members and our faculty advisor.

2. Understanding the problem domain

There was some confusion in regards to the project outline specifics. We were able to resolve any questions that we had in our 1st presentation alongside with one on one meetings we had with both our industry and faculty mentors. Their ongoing feedback in the meetings was very helpful since it allows our team to be on track of the tasks we were working on and maintaining an overall consistent progress of the project.

3. Version control management

As we worked on the code, we did run through some version control problems. We noticed that the code ran well in Python versions 3.6, 3.8 and below. For Python version 3.8 we needed to uncheck for future integration to get it to run. We saw ourselves in needing to make changes for Python 3.9 and above. Since we were all using different versions, we decided to run our application in various different versions of Python to see how it ran, giving us a better idea how to program the code to ensure that the user can access and run it.

4. Understanding the main program of the project

It really took time to understand our program's set up since all of us had different programming approaches when it came down to coding. We had to review and study each other's code to understand how it worked so that we know what were the tasks that we needed to add when working on the same program. Alongside, we were learning new coding approaches and techniques in Python as we went about designing the program.

5. Small cycle releases

It's challenging to ensure that each release adds value and doesn't sacrifice the quality for speed. Testing can be a real time sink, and technical debt can pile up quickly. With each release, we needed to make sure that it actually adds value and doesn't just introduce small incremental changes. But the benefits of frequent feedback and manageable development chunks make it worth it in the end. By getting feedback from users and stakeholders on each release, we can stay more aligned with their needs and make sure were on the right track. Plus, breaking the development process down into small, manageable chunks helps us stay focused and make progress quickly.

~~6. Active User Involvement~~

For our project, we decided to have user involvement throughout most of the development of the application.

This interaction was both beneficial and a challenge since it allowed our team to see how the program was functioning and to see the necessary changes and edits that needed to be done consistently to make in the code to ensure that the user can access and run it. It added additional workload but it paid off because we were able to get a more realistic view of how user interaction with the software was taking place.

10.0 System Manuals

10.1.1 How to set up development environment

1. Make sure python version 3.9 is downloaded onto your directory
2. Open Terminal or Command Prompt
3. git clone Windows OS Version Application
4. git clone Mac OS Version Application
5. After it is cloned(downloaded) install packages
6. Or download the application as a zip file and open the zip file
7. In the command 'cd' to your working directory: '**cd TARGEST.Windows.Final-1.4**' or '**cd TARGEST.MAC.Final-1.4**'
8. Pip install requirements.txt
9. Once packages are installed, the program is ready to run.
10. Then type, "ls" to make sure all files have been cloned.
11. Finally, run code by typing '**python main.py'** or '**python3 main.py**'

    or **./main** to run the application.

11.0 System Testing

We designed a total of 5 test cases to thoroughly test our code. Prior to testing, we refactored the code significantly, as there was a lot of unnecessary and untestable code present. We reduced the size of the code file and improved its overall structure to make it easier to test.

```
class TestGenerateReport(unittest.TestCase):
    Targest2.generateReport()

    def test_filename(self):
        expected_output = ["HDS_new_pump", "HRS_new_pump","HTP_new_pump", "HTR_new_pump", "PRS_new_pump", "RiskAnalysis_Pump",
                           "SDS_New_pump_x04", "SRS_ACE_Pump_X01", "SRS_BolusCalc_Pump_X04", "SRS_DosingAlgorithm_X03",
                           "SVaP_new_pump", "SVaTR_new_pump", "SVeTR_new_pump", "URS_new_pump"]

        self.assertEqual(Targest2.filename_collection, expected_output)
```

This test involves the creation of a class that performs two testing functions after generating a report. The first function retrieves a collection of file names and compares them with the expected output list provided in the image, thereby ensuring that all files located in the specified path are correctly identified.

```
def test_parent_tags(self):
    expected_output = [['PUMP:HRD:100', 'PUMP:HRD:105', 'PUMP:HRD:1000', 'PUMP:HRD:3330', 'PUMP:HRD:3350'],
                       ['PUMP:HRS:100', 'PUMP:HRS:105', 'PUMP:HRS:1000', 'PUMP:HRS:3330', 'PUMP:HRS:3340',
                        'PUMP:HRS:3350'], ['PUMP:HTP:100', 'PUMP:HTP:200', 'PUMP:HTP:300', 'PUMP:HTP:400',
                        'PUMP:HTP:500', 'PUMP:HTP:1100', 'PUMP:HTP:1200', 'PUMP:HTP:1300', 'PUMP:HTP:1400',
                        'PUMP:HTP:1500'], ['PUMP:HTR:100', 'PUMPHTR:200', 'PUMP:HTR:300', 'PUMP:HTR:400',
                        'PUMP:HTR:500', 'PUMP:HTR:1100', 'PUMP:HTR:1200', 'PUMP:HTR:1300', 'PUMP:HTR:1400',
                        'PUMP:HTR:1500'],['PUMP:PRS:1', 'PUMP:PRS:2', 'PUMP:PRS:3', 'PUMP:PRS:4', 'PUMP:PRS:5',
                        'PUMP:PRS:8', 'PUMP:PRS:10', 'PUMP:PRS:100', 'PUMP:PRS:105', 'PUMP:PRS:1000',
                        'PUMP:PRS:3330', 'PUMP:PRS:3340', 'PUMP:PRS:3350', 'PUMP:PRS:4000'],['PUMP:RISK:10',
                        'PUMP:RISK:20', 'PUMP:RISK:30', 'PUMP:RISK:40', 'PUMP:RISK:50'], ['PUMP:SDS:10',
                        'PUMP:SDS:20', 'PUMP:SDS:30', 'PUMP:SDS:40', 'PUMP:SDS:50', 'PUMP:SDS:60', 'PUMP:SDS:70'],
                       ['ACE:SRS:1', 'ACE:SRS:2', 'ACE:SRS:5', 'ACE:SRS:6', 'ACE:SRS:10', 'ACE:SRS:100'],
                       ['BOLUS:SRS:1', 'BOLUS:SRS:2', 'BOLUS:SRS:5', 'BOLUS:SRS:6', 'BOLUS:SRS:8', 'BOLUS:SRS:12'],
```

This test is focused on the list of lists provided in the first document. Specifically, it examines all the parent tags that are correctly identified and reads their corresponding child elements. The test then compares the resulting list of parent elements to the expected output.

```
def test_orphan_child_tags(self):
    expected_output = ['PUMP:URS:1', 'PUMP:URS:3', 'PUMP:URS:8', 'PUMP:URS:10', 'PUMP:URS:100', 'PUMP:URS:103',
                       'PUMP:URS:1000', 'PUMP:URS:3330', 'PUMP:URS:3350', 'PUMP:URS:4000']

    self.assertEqual(Targest2.orphanChild, expected_output)
```

Orphan tags list show expected output and file it is looking at

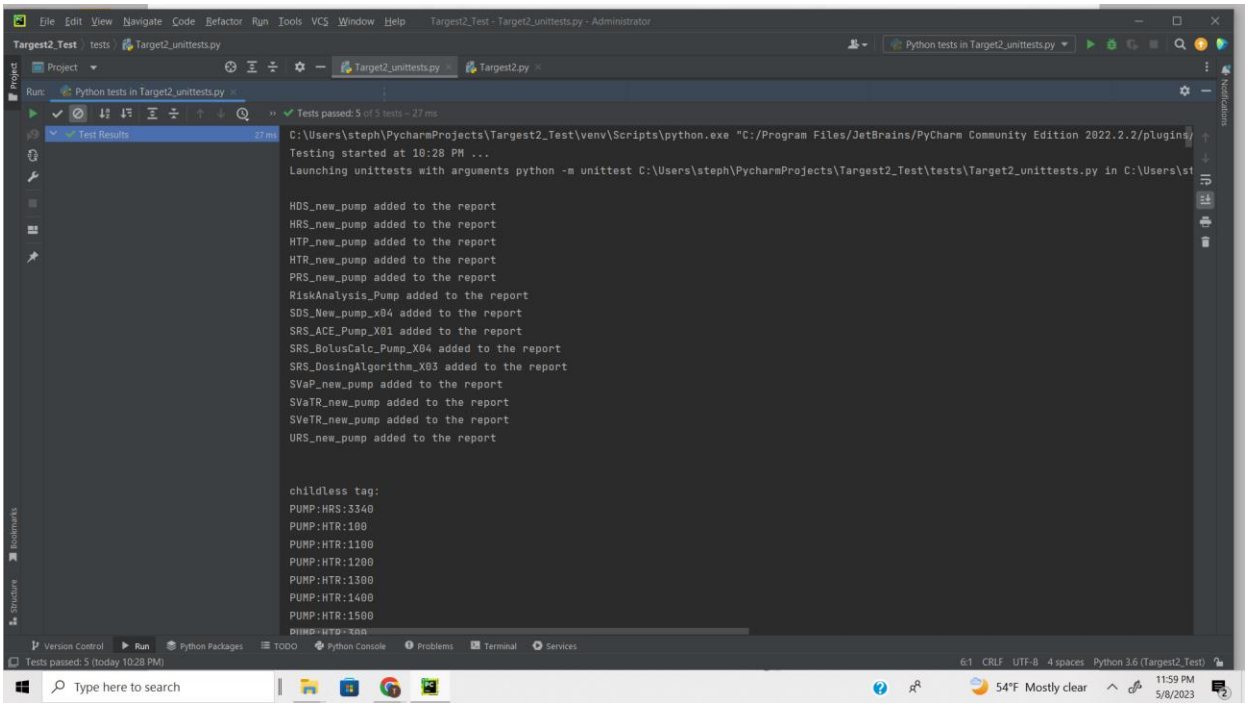
```
def test_child_tags(self):
    expected_output = [['PUMP:HRD:100', 'PUMP:HRD:105', 'PUMP:HRD:1000', 'PUMP:HRD:3330', 'PUMP:HRD:3350'],
                       ['PUMP:HRS:100', 'PUMP:HRS:105', 'PUMP:HRS:1000', 'PUMP:HRS:3330', 'PUMP:HRS:3340',
                        'PUMP:HRS:3350'], ['PUMP:HTP:100', 'PUMP:HTP:200', 'PUMP:HTP:300', 'PUMP:HTP:400',
                        'PUMP:HTP:500', 'PUMP:HTP:1100', 'PUMP:HTP:1200', 'PUMP:HTP:1300', 'PUMP:HTP:1400',
                        'PUMP:HTP:1500'], ['PUMP:HTR:100', 'PUMPHTR:200', 'PUMP:HTR:300', 'PUMP:HTR:400',
                        'PUMP:HTR:500', 'PUMP:HTR:1100', 'PUMP:HTR:1200', 'PUMP:HTR:1300', 'PUMP:HTR:1400',
                        'PUMP:HTR:1500'], ['PUMP:PRS:1', 'PUMP:PRS:2', 'PUMP:PRS:3', 'PUMP:PRS:4', 'PUMP:PRS:5',
                        'PUMP:PRS:8', 'PUMP:PRS:10', 'PUMP:PRS:100', 'PUMP:PRS:105', 'PUMP:PRS:1000',
                        'PUMP:PRS:3330', 'PUMP:PRS:3340', 'PUMP:PRS:3350', 'PUMP:PRS:4000'], ['PUMP:RISK:10',
                        'PUMP:RISK:20', 'PUMP:RISK:30', 'PUMP:RISK:40', 'PUMP:RISK:50'], ['PUMP:SDS:10',
```

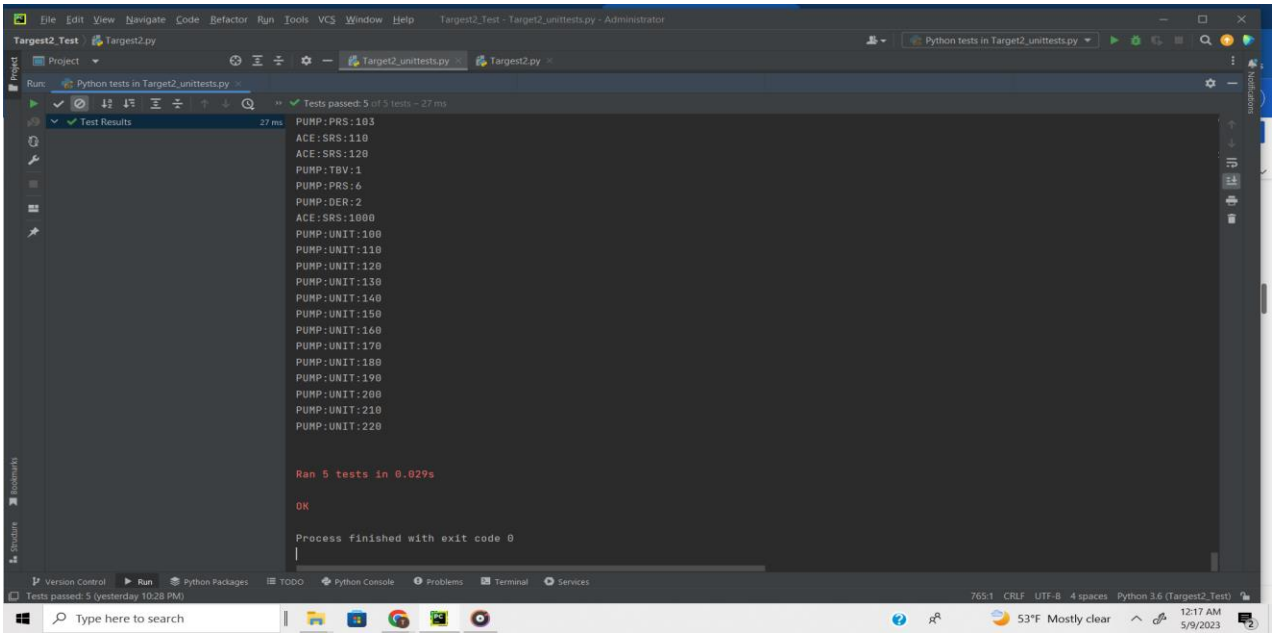
Leading tags list that show expected output and correspond documents

```
def test_childless_tags(self):
    expected_output = ['PUMP:HRS:3340', 'PUMP:HTR:100', 'PUMP:HTR:1100', 'PUMP:HTR:1200', 'PUMP:HTR:1300',
                       'PUMP:HTR:1400', 'PUMP:HTR:1500', 'PUMP:HTR:300', 'PUMP:HTR:400', 'PUMP:HTR:500',
                       'PUMP:INS:100', 'PUMP:INS:110', 'PUMP:INS:120', 'PUMP:INS:130', 'PUMP:INS:140',
                       'PUMP:INS:150', 'PUMP:INS:160', 'PUMP:INS:170', 'PUMP:INS:180', 'PUMP:INS:190',
                       'PUMP:INS:200', 'PUMP:INS:210', 'PUMP:INS:220', 'PUMP:PRS:2', 'PUMP:PRS:3340',
                       'PUMP:SDS:10', 'PUMP:SDS:20', 'PUMP:SDS:30', 'PUMP:SDS:40', 'PUMP:SDS:50', 'PUMP:SDS:60',
                       'PUMP:SDS:70', 'PUMP:SVATR:100', 'PUMP:SVATR:200', 'PUMP:SVATR:300', 'PUMP:SVATR:400',
                       'PUMP:SVATR:500', 'PUMP:UT:100', 'PUMP:UT:110', 'PUMP:UT:120', 'PUMP:UT:130',
                       'PUMP:UT:140', 'PUMP:UT:150', 'PUMP:UT:160', 'PUMP:UT:170', 'PUMP:UT:180', 'PUMP:UT:190',
                       'PUMP:UT:200', 'PUMP:UT:210', 'PUMP:UT:220', 'PUMPHTR:200']
    self.assertEqual(Targest2.childless, expected_output)
```

Childless tags list show expected output and looks into variable called child's list

Test run and output successes





12.0 References

Appendix

U (generated from Use Case Printout on Capstone)

4.1.3.2 User Scenarios (Use Cases):

Table 4.1. Use Case Index Table

Table 4.1. Use Case Index Table

Project Name: Requirements Tracing/Management Software				
Use Case ID	**Use Case Name**	**Level**	**Author**	**Version**
UC-001	Case 1	Primary task	Adrian Bernardino	0.4
UC-002	Case 2	Primary task	Adrian Bernardino	0.5
UC-003	Case 4	Summary	Jan Haug	0.4
UC-004	Case 7	Summary	Adrian Bernardino	0.2
UC-005	Case 3	Primary task	Stephania Rey	0.2
UC-006	Case 5	Primary task	Stephania Rey	0.1
UC-007	Case 6	Summary	Stephania Rey	0.1
Acknowledgment: Generated from the CapStone process management system ©2023				

Table 4.2. Use Case UC-001:

Project Name:	**Requirements Tracing/Management Software**
Use Case ID:	UC-001
Use Case Name:	Case 1
User Goal:	Should expect the user to be able to open the software on both Mac-OS and Windows.
Scope:	Targest
Level:	Primary task
Relevant User Reqs:	UF-A
Relevant System Reqs:	SF-A-01
Primary Actor:	User
Precondition:	When the user runs the software.
Minimal Guarantee:	The user is not able to access the software.
Success Guarantee:	The user is able to open using platforms such as Mac-OS and Windows.
Trigger:	Opening the software
Success Scenario:	**Step Actions**
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Table 4.3. Use Case UC-002:

Table 4.3. Use Case UC-002

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-002
Use Case Name:	Case 2
User Goal:	The system should allow the user to enter certain instructions that will facilitate certain tasks.
Scope:	Targest
Level:	Primary task
Relevant User Reqs:	UF-B,UF-C
Relevant System Reqs:	SF-A-01,SF-B-02,SF-B-03,SF-C-01,SF-D-01,SF-E-01,SF-F-01,SF-G-01,SF-H-01
Primary Actor:	User
Precondition:	Starts when the user runs the software
Minimal Guarantee:	The user will not have access to enter certain instructions
Success Guarantee:	The user will be able to display tasks upon entering
Trigger:	Once the user opens the software
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

Table 4.4. Use Case UC-003:

Table 4.4. Use Case UC-003

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-003
Use Case Name:	Case 4
User Goal:	Parse documents for tags
Scope:	Targest
Level:	Summary
Relevant User Reqs:	UF-B
Relevant System Reqs:	SF-B-01,SF-B-02,SF-B-03,SF-C-01,SF-D-01,SF-E-01,SF-F-01,SF-G-01,SF-H-01
Primary Actor:	User
Precondition:	The software is running
Minimal Guarantee:	The user would get unfinished reports
Success Guarantee:	The user would get a parsed document
Trigger:	It will starts when the software runs the execution of the code
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

Table 4.5. Use Case 5 UC-004:

Table 4.5. Use Case UC-004

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-004
Use Case Name:	Case 7
User Goal:	The system should link documents based on tags and display linkage.
Scope:	Targest
Level:	Summary
Relevant User Reqs:	UF-B
Relevant System Reqs:	SF-B-02,SF-B-03
Primary Actor:	User
Precondition:	Once the document is parsed it will link documents.
Minimal Guarantee:	Unsuccessful linkage between the documents causing false data output.
Success Guarantee:	A display of linkage between documents that were chosen.
Trigger:	Parsing documents first then display linkage of documents.
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

Table 4.6. Use Case 5 UC-005:

Table 4.6. Use Case UC-005

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-005
Use Case Name:	Case 3
User Goal:	Match documents based on tags
Scope:	Targest
Level:	Primary task
Relevant User Reqs:	UF-D,UF-E,UF-F,UF-G,UF-H
Relevant System Reqs:	SF-B-01,SF-B-02,SF-B-03
Primary Actor:	User
Precondition:	The software is running
Minimal Guarantee:	The user will not be able to match documents based on tags
Success Guarantee:	The user would be able to match documents based on tags
Trigger:	Documents are matched and prompts window showing confirmation
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

Table 4.7. Use Case UC-006:

Table 4.7. Use Case UC-006

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-006
Use Case Name:	Case 5
User Goal:	Match a tag with given set of tags
Scope:	Targest
Level:	Primary task
Relevant User Reqs:	UF-B
Relevant System Reqs:	SF-B-01,SF-B-02,SF-B-03
Primary Actor:	User
Precondition:	The software is running
Minimal Guarantee:	The user is not able to match a tag with given set of tags
Success Guarantee:	The user matches a tag with given set of tags
Trigger:	Verification of tag match
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

Table 4.8. Use Case UC-007:

Table 4.8. Use Case UC-007

Project Name:	Requirements Tracing/Management Software
Use Case ID:	UC-007
Use Case Name:	Case 6
User Goal:	Search other documents for same tag
Scope:	Targest
Level:	Summary
Relevant User Reqs:	UF-B
Relevant System Reqs:	SF-B-02,SF-B-03
Primary Actor:	User
Precondition:	The software is running
Minimal Guarantee:	The user is not able to search other documents with the same tag
Success Guarantee:	The user gets confirmation of search documents with same tag
Trigger:	Opening the software
Success Scenario:	Step Actions
Extensions:	Branching Scenarios
Acknowledgment: Generated from the CapStone process management system ©2023	

R (generated from Requirements Printout on Capstone)

4.1.3.3 User Functional Requirements: UF-

Table 4.9. User Functional Requirements: UF-A

Table 4.9. User Functional Requirements: UF-A

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-A	Type	Functional	Non-Functional	
Creation:	Dec 03 2022 02:57 PM	User	☒	☐	
Modification:	May 08 2023 02:45 PM	System	☐	☐	
Description:	The system shall allow the user to run the software on both OS and Windows.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-A-01				
Justify why UF-A can be completely covered by SF-A-01	UF-A can be completely covered by SF-A-01 since our software works on both MacOS and Windows OS.				
Traceability:	Use cases cf.	UC-001			
	Test cases cf.	TC-002, TC-005			
Acknowledgment	Generated from the CapStone Process Management System ©2023				

Table 4.10. User Functional Requirements: UF-B

Table 4.10. User Functional Requirements: UF-B

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-B	Type	Functional	Non-Functional	
Creation:	Sep 26 2022 08:20 PM	User	☒	☐	
Modification:	May 08 2023 02:48 PM	System	☐	☐	
Description:	The user shall be able to experience an instantaneous interface with fast runtime regardless of the amount of documentation generated.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-B-01, SF-B-02, SF-B-03				
Justify why UF-B can be completely covered by SF-B-01, SF-B-02, SF-B-03	UF-B can be completely covered by SF-B-01, SF-B-02, SF-B-03 since we have fulfilled the features needed in our application to give the user the desired experience.				
Traceability:	Use cases cf.	UC-002, UC-003, UC-004, UC-006, UC-007			
	Test cases cf.	TC-001, TC-002, TC-004, TC-005			
Acknowledgment	Generated from the CapStone Process Management System ©2023				

Table 4.11. User Functional Requirements: UF-C

Table 4.11. User Functional Requirements: UF-C

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-C	Type	Functional	Non-Functional	
Creation:	Dec 03 2022 03:04 PM	User	☒	☐	
Modification:	Dec 03 2022 03:09 PM	System	☐	☐	
Description:	The software support the user generating multiple reports such as lists of all tags used.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-C-01				
Justify why UF-C can be completely covered by SF-C-01	UF-C can be completely covered by SF-C-01 because if the system properly executes our tests, then it should generate reports desired.				
Traceability:	Use cases cf.	UC-002			
	Test cases cf.	TC-001, TC-002			
Acknowledgment	Generated from the CapStone Process Management System ©2023				

Table 4.12. User Functional Requirements: UF-D

Table 4.12. User Functional Requirements: UF-D

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-D			Type: User / System	Functional: ☒ / ☐ — Non-Functional: ☐ / ☐
Creation:	Dec 03 2022 03:11 PM				
Modification:	May 08 2023 02:50 PM				
Description:	The software should support the user generating multiple reports such as lists of all tags that have no children.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-D-01				
Justify why UF-D can be completely covered by SF-D-01	UF-D can be completely covered by SF-F-01 since our software is able find childless tags that are necessary to catch since it is being confirmed that the user added the tags and requirements wrong.				
Traceability:	Use cases cf.	UC-005			
	Test cases cf.	TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.13. User Functional Requirements: UF-E

Table 4.13. User Functional Requirements: UF-E

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-E			Type: User / System	Functional: ☒ / ☐ — Non-Functional: ☐ / ☐
Creation:	Dec 03 2022 03:13 PM				
Modification:	May 08 2023 02:48 PM				
Description:	The software should support the user generating multiple reports such as lists of tags that have no parents.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-E-01				
Justify why UF-E can be completely covered by SF-E-01	This is displayed in the generation of the parentless tags report.				
Traceability:	Use cases cf.	UC-005			
	Test cases cf.	TC-001			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.14. User Functional Requirements: UF-F

Table 4.14. User Functional Requirements: UF-F

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-F			Type: User / System	Functional: ☒ / ☐ — Non-Functional: ☐ / ☐
Creation:	Dec 03 2022 03:15 PM				
Modification:	May 08 2023 02:42 PM				
Description:	The software should support the user generating multiple reports such as lists of tags that are not tested.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-F-01				
Justify why UF-F can be completely covered by SF-F-01	This is addressed in the print out document of TBV.				
Traceability:	Use cases cf.	UC-005			
	Test cases cf.	TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.15. User Functional Requirements: UF-G

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-G			Type: User / System	Functional: ☒ / ☐ — Non-Functional: ☐ / ☐
Creation:	Dec 03 2022 03:16 PM				
Modification:	May 08 2023 02:45 PM				
Description:	The software should support the user generating multiple reports such as lists of duplicate tags.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-G-01				
Justify why UF-G can be completely covered by SF-G-01	Duplicate tags are generated but quickly resolved to demonstrate most current state.				
Traceability:	Use cases cf.	UC-005			
	Test cases cf.	TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.16. User Functional Requirements: UF-H

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UF-H			Type: User / System	Functional: ☒ / ☐ — Non-Functional: ☐ / ☐
Creation:	Dec 03 2022 03:17 PM				
Modification:	May 08 2023 02:46 PM				
Description:	The software should support the user generating multiple reports such as texts reports showing parents adjacent to children.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-H-01				
Justify why UF-H can be completely covered by SF-H-01	This is displayed in the generation of the All Parents/Child Tags documents showing their relationship to each other				
Traceability:	Use cases cf.	UC-005			
	Test cases cf.	TC-001, TC-004			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

4.1.4. User Non-Functional Requirements: UO-

Table 4.17. User Non-Functional Requirements: UP-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UP-01			Type / Functional / Non-Functional	
Creation:	Oct 22 2022 02:37 PM			User ☐ / ☒	
Modification:	Oct 22 2022 02:43 PM			System ☐ / ☐	
Description:	The system should support cross platforms and should be able to be used on a computer device.			Product (sub-type below)	
				Usability Requirements	
Priority:	✓ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SP-01-01				
Justify why UP-01 can be completely covered by SP-01-01	UP-01 can be completely covered by SP-01-01 because of the standard that is being established which allows for us to reach more users.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-001, TC-002, TC-004, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.18. User Non-Functional Requirements: UO-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	UO-01			Type / Functional / Non-Functional	
Creation:	Sep 26 2022 08:42 PM			User ☐ / ☒	
Modification:	Sep 26 2022 08:55 PM			System ☐ / ☐	
Description:	The application shall consist of 5 documents, 5 embed tags, and 5 test cases.			Organizational (sub-type below)	
				Development Requirements	
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Refined Into:	SO-01-01				
Justify why UO-01 can be completely covered by SO-01-01	UO-01 can be completely covered by SO-01-01 since the application will have external tools supporting desired operations.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-002			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

4.2.1.1. System Functional Requirements: SF-

Table 4.19. System Functional Requirements: SF-A-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-A-01			Type / Functional / Non-Functional	
Creation:	Dec 03 2022 04:30 PM			User ☐ / ☐	
Modification:	Dec 03 2022 04:32 PM			System ☒ / ☐	
Description:	The system shall be able to run on multiple platforms.				
Priority:	✓ Highest	High	Medium	Low	Lowest
This Req. is Engineered From:	UF-A				
Justify why meeting SF-A-01 can contribute to the fulfilment of UF-A	SF-A-01				
Traceability:	Use cases cf.	UC-001, UC-002			
	Test cases cf.	TC-001, TC-002, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.20. System Functional Requirements: SF-B-01

Project Name:	Requirements Tracing/Management Software					
Requirement #:	SF-B-01			Type	Functional	Non-Functional
Creation:	Sep 26 2022 08:27 PM			User	☐	☐
Modification:	Sep 26 2022 08:33 PM			System	☒	☐
Description:	The system shall have an optimum graphical user-interface.					
Priority:	Highest	✓ High	Medium	Low	Lowest	
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-01 can contribute to the fulfilment of UF-B	SF-B-01 can contribute to the fulfillment of UF-B because if user-interface is optimum, then this allows for a successful and fast user experience with the software.					
Traceability:	Use cases cf.	UC-003, UC-005, UC-006				
	Test cases cf.	TC-003, TC-004, TC-005				
Acknowledgment	*Generated from the CapStone Process Management System ©2023*					

Table 4.21. System Functional Requirements: SF-B-02

Project Name:	Requirements Tracing/Management Software					
Requirement #:	SF-B-02			Type	Functional	Non-Functional
Creation:	Sep 27 2022 08:54 PM			User	☐	☐
Modification:	Sep 27 2022 09:09 PM			System	☒	☐
Description:	The system shall display the number of errors.					
Priority:	Highest	✓ High	Medium	Low	Lowest	
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-02 can contribute to the fulfilment of UF-B	SF-B-02 can contribute to the fulfillment of UF-B because if errors are displayed then the user is satisfied.					
Traceability:	Use cases cf.	UC-002, UC-003, UC-004, UC-005, UC-006, UC-007				
	Test cases cf.	TC-002, TC-005				
Acknowledgment	*Generated from the CapStone Process Management System ©2023*					

Table 4.22. System Functional Requirements: SF-B-03

Project Name:	Requirements Tracing/Management Software					
Requirement #:	SF-B-03			Type	Functional	Non-Functional
Creation:	Sep 27 2022 09:02 PM			User	☐	☐
Modification:	Sep 27 2022 09:09 PM			System	☒	☐
Description:	The system shall display number of successes.					
Priority:	Highest	✓ High	Medium	Low	Lowest	
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-03 can contribute to the fulfilment of UF-B	SF-B-03 can contribute to the fulfillment of UF-B because if the successes are displayed, then the user is satisfied.					
Traceability:	Use cases cf.	UC-002, UC-003, UC-004, UC-005, UC-006, UC-007				
	Test cases cf.	TC-002, TC-005				
Acknowledgment	*Generated from the CapStone Process Management System ©2023*					

Table 4.23. System Functional Requirements: SF-C-01

Project Name:	Requirements Tracing/Management Software					
Requirement #:	SF-C-01			Type	Functional	Non-Functional
Creation:	Dec 03 2022 03:07 PM			User	☐	☐
Modification:	Dec 03 2022 03:08 PM			System	☒	☐
Description:	The system shall generate a lists of all tags using xlwings.					
Priority:	Highest	✓ High	Medium	Low	Lowest	
This Req. is Engineered From:	UF-C					
Justify why meeting SF-C-01 can contribute to the fulfilment of UF-C	SF-C-01 can contribute to the fulfillment of UF-C by using xlwings to extract tags onto an excel sheet.					
Traceability:	Use cases cf.	UC-002, UC-003				
	Test cases cf.	TC-001, TC-002				
Acknowledgment	*Generated from the CapStone Process Management System ©2023*					

Table 4.24. System Functional Requirements: SF-D-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-D-01			Type	Functional	Non-Functional
Creation:	Dec 03 2022 03:21 PM			User	☐	☐
Modification:	Dec 03 2022 03:22 PM			System	☒	☐
Description:	The system shall generate a lists of all tags that have no children.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-D				
Justify why meeting SF-D-01 can contribute to the fulfilment of UF-D	SF-D-01 can contribute to the fulfillment of UF-D by using docx and xlwings to extract tags.				
Traceability:	Use cases cf.	UC-002, UC-003			
	Test cases cf.	TC-001			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.25. System Functional Requirements: SF-E-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-E-01			Type	Functional	Non-Functional
Creation:	Dec 03 2022 03:28 PM			User	☐	☐
Modification:	Dec 03 2022 03:30 PM			System	☒	☐
Description:	The system shall generate a list of tags that have no parents using docx or xlwings.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-E				
Justify why meeting SF-E-01 can contribute to the fulfilment of UF-E	SF-E-01 can contribute to the fulfillment of UF-E by using docx or xlwings to extract tags.				
Traceability:	Use cases cf.	UC-002, UC-003			
	Test cases cf.	TC-001			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.26. System Functional Requirements: SF-F-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-F-01			Type	Functional	Non-Functional
Creation:	Dec 03 2022 03:31 PM			User	☐	☐
Modification:	Dec 03 2022 03:34 PM			System	☒	☐
Description:	The system shall generate lists of tags that are not tested using docx or xlwings.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-F				
Justify why meeting SF-F-01 can contribute to the fulfilment of UF-F	SF-F-01 can contribute to the fulfillment of UF-F by using docx or xlwings to extract tags that are not tested.				
Traceability:	Use cases cf.	UC-002, UC-003			
	Test cases cf.	TC-001, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.27. System Functional Requirements: SF-G-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-G-01			Type	Functional	Non-Functional
Creation:	Dec 03 2022 03:55 PM			User	☐	☐
Modification:	Dec 03 2022 03:57 PM			System	☒	☐
Description:	The system shall generate a lists of duplicate tags				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-G				
Justify why meeting SF-G-01 can contribute to the fulfilment of UF-G	SF-G-01 can contribute to the fulfillment of UF-G by using docx or xlwings to extract tags.				
Traceability:	Use cases cf.	UC-002, UC-003			
	Test cases cf.	TC-001			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.28. System Functional Requirements: SF-H-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SF-H-01	Type	Functional	Non-Functional	
Creation:	Dec 03 2022 03:57 PM	User	☐	☐	
Modification:	Dec 03 2022 03:59 PM	System	☒	☐	
Description:	The system shall generate a lists of texts showing parents adjacent to children.				
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-H				
Justify why meeting SF-H-01 can contribute to the fulfilment of UF-H	SF-H-01 can contribute to the fulfillment of UF-H by using docx or xlwings to extract text showing adjacent tags.				
Traceability:	Use cases cf.	UC-002, UC-003			
	Test cases cf.	TC-001, TC-004			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

4.2.2. System Non-Functional Requirements: SO-

Table 4.29. System Non-Functional Requirements: SP-01-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SP-01-01	Type	Functional	Non-Functional	
Creation:	Oct 22 2022 02:37 PM	User	☐	☐	
Modification:	Oct 22 2022 02:39 PM	System	☐	☒	
Description:	The system shall run on both OS and Windows platforms.	Product (sub-type below)			
		Usability Requirements			
Priority:	✓ Highest	High	Medium	Low	Lowest
This Req. is Engineered From:	UP-01				
Justify why meeting SP-01-01 can contribute to the fulfilment of UP-01	SP-01-01 can contribute to the fulfillment of UP-01 because the user will be able to access this software on either platforms.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-001, TC-002, TC-003, TC-004, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

Table 4.30. System Non-Functional Requirements: SO-01-01

Project Name:	Requirements Tracing/Management Software				
Requirement #:	SO-01-01	Type	Functional	Non-Functional	
Creation:	Sep 26 2022 08:48 PM	User	☐	☐	
Modification:	Sep 26 2022 08:53 PM	System	☐	☒	
Description:	The system supports external tools such as Pydoc, xlwings, and PUnit or ProUnit.	Organizational (sub-type below)			
		Development Requirements			
Priority:	Highest	✓ High	Medium	Low	Lowest
This Req. is Engineered From:	UO-01				
Justify why meeting SO-01-01 can contribute to the fulfilment of UO-01	SO-01-01 can contribute to the fulfillment of UO-01 because of the supportive software tools that will allow the application to use documents, embed tags, and test cases.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-001, TC-002, TC-004, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2023*				

4.3 Requirements Trace Table:

Table 4.31. Mapping from user requirements to system requirements

Project Name: Requirements Tracing/Management Software			
User Requirements		System Requirements	
Req ID	Description	Req ID	Description
UF-A	The system shall allow the user to run the software on both OS and Windows.	SF-A-01	The system shall be able to run on multiple platforms.
UF-B	The user shall be able to experience an instantaneous interface with fast runtime regardless of the amount of documentation generated.	SF-B-01	The system shall have an optimum graphical user-interface.
		SF-B-02	The system shall display the number of errors.
		SF-B-03	The system shall display number of successes.
UF-C	The software support the user generating multiple reports such as lists of all tags used.	SF-C-01	The system shall generate a lists of all tags using xlwings.
UF-D	The software should support the user generating multiple reports such as lists of all tags that have no children.	SF-D-01	The system shall generate a lists of all tags that have no children.
UF-E	The software should support the user generating multiple reports such as lists of tags that have no parents.	SF-E-01	The system shall generate a list of tags that have no parents using docx or xlwings.
UF-F	The software should support the user generating multiple reports such as lists of tags that are not tested.	SF-F-01	The system shall generate lists of tags that are not tested using docx or xlwings.
UF-G	The software should support the user generating multiple reports such as lists of duplicate tags.	SF-G-01	The system shall generate a lists of duplicate tags
UF-H	The software should support the user generating multiple reports such as texts reports showing parents adjacent to children.	SF-H-01	The system shall generate a lists of texts showing parents adjacent to children.
UO-01	The application shall consist of 5 documents, 5 embed tags, and 5 test cases.	SO-01-01	The system supports external tools such as Pydoc, xlwings, and PUnit or ProUnit.
UP-01	The system should support cross platforms and should be able to be used on a computer device.	SP-01-01	The system shall run on both OS and Windows platforms.
Acknowledgment: Generated from the CapStone process management system ©2023			

T (Code Source)

Main.py

```
import Gui # Imports the Gui.py file


if __name__ == "__main__" :
    Gui.GUI1() # Calls the function GUI1() from the Gui.py file
```

TARGEST.py

```
import openpyxl
import tkinter as tk
from tkinter import ttk
from tkinter.scrolledtext import ScrolledText
from tkinter import *


import Targest2


def process_tree(ws, row, col, parent=None):

family_tree = []
current_node = ws.cell(row=row, column=col).value

 if current_node is None or current_node.lower() == 'separator':
    return family_tree

  family_tree.append((current_node, col - 1, parent))
  child_row = row + 1
  child_col = col + 1


  while ws.cell(row=child_row, column=child_col).value is not None:
    child_node = ws.cell(row=child_row, column=child_col).value
    if child_node.lower() != 'separator':
        child_tree = process_tree(ws, child_row, child_col, current_node)
        family_tree.extend(child_tree)
    else:
        break
    child_row += 1


 sibling_col = col
 sibling_row = row + 1
 while ws.cell(row=sibling_row, column=sibling_col).value is not None:
```

```
        sibling_node = ws.cell(row=sibling_row, column=sibling_col).value
        if sibling_node.lower() != 'separator':
            sibling_tree = process_tree(ws, sibling_row, sibling_col, parent)
            family_tree.extend(sibling_tree)
        sibling_row += 1


    return family_tree



def build_tree_structure(tree):
    tree_structure = {}
    for line in tree:
        if len(line) != 3:
            continue


        node, col, parent = line
        level = col - 1  # Change this line
        if level not in tree_structure:
            tree_structure[level] = []
        tree_structure[level].append({"node": node, "parent": parent})


    return tree_structure



def display_tree3(tree_structure):
    if not tree_structure or 0 not in tree_structure:
        return ""


    tree_str = ""
    root_node = tree_structure[0][0]
```

```
    def print_node(node, level, last_child=False, parent=None):
        nonlocal tree_str
        indent = "│   " * level if level > 0 else ""
        indent += " └─ " if last_child else " ├─ "
        tree_str += indent + node["node"] + "\n"
        children = tree_structure.get(level + 1, [])
        children = [child for child in children if child["parent"] == node["node"]]
        for i, child in enumerate(children):
            print_node(child, level + 1, i == len(children) - 1, node["node"])


    print_node(root_node, 0)
    return tree_str




def text3(window):
    family_trees = Targest2.guiTree()
    print("text3 function called")
    #print(family_trees)
    data_string = convert_to_string(family_trees)
    print(data_string)
    #Targest2.TBDReport.add_paragraph(data_string)
    #Targest2.TBDReport.save('TBDReport.docx')
# Create a vertical scrollbar
    scrollbar_y = Scrollbar(window, orient=VERTICAL)
    scrollbar_y.pack(side=RIGHT, fill=Y)


    # Create a horizontal scrollbar
    scrollbar_x = Scrollbar(window, orient=HORIZONTAL)
    scrollbar_x.pack(side=BOTTOM, fill=X)
```

```
    # Create a ScrolledText widget and configure its scrollbars
    scrolled_text_box = ScrolledText(window, wrap=NONE, xscrollcommand=scrollbar_x.set,
yscrollcommand=scrollbar_y.set, height=15, width=47)
    scrolled_text_box.place(x=566, y=240)
    scrolled_text_box.configure(bg='grey', fg='white')


    # Configure the scrollbars
    scrollbar_x.config(command=scrolled_text_box.xview)
    scrollbar_y.config(command=scrolled_text_box.yview)
    #scrolled_text_box.delete(1.0, tk.END)  # Clear the scrolltext box
    #scrolled_text_box.insert(tk.END, data_string)  # Insert the converted string
    #scrolled_text_box.insert(tk.END, f"Family Tree {i}:\n")
    #scrolled_text_box.insert(tk.END, "------------\n")
    #scrolled_text_box.insert(tk.END, display_result + "\n")
    scrolled_text_box.insert(tk.END, data_string)  # Insert the converted string
    #for i, tree in enumerate(family_trees, 1):
     #   display_result = display_tree2(tree)
      #  if display_result:
       #     scrolled_text_box.insert(tk.END, f"Family Tree {i}:\n")
        #    scrolled_text_box.insert(tk.END, "------------\n")
         #   #scrolled_text_box.insert(tk.END, display_result + "\n")
          #  scrolled_text_box.insert(tk.END, data_string)  # Insert the converted string



def display_tree2(tree):
    if not tree or tree[0][0].lower() == 'separator':
        return ""


    tree_dict = {}
    for node, level, parent in tree:
        if level in tree_dict:
            tree_dict[level].append((node, parent))
```

```
        else:
            tree_dict[level] = [(node, parent)]


    tree_str = ""


    def print_node(node, level, last_child=False, parent=None):
        nonlocal tree_str
        indent = "  " * level + (" └─ " if last_child else " ├─ ")
        tree_str += indent + node + "\n"
        children = tree_dict.get(level + 1, [])
        children = [child for child, child_parent in children if child_parent == node]
        for i, child in enumerate(children):
            print_node(child, level + 1, i == len(children) - 1, node)


    root_node = tree_dict[0][0][0]
    print_node(root_node, 0)
    return tree_str


def convert_to_string(data):
    result = ""
    for inner_list in data:
        for item in inner_list:
            result += item
        result += "\n"
    return result
```

TARGEST2.py

```
# This program will search for red colored tags to get the information it needs
#
# Instructions on how to use the program:
# 1. run program
# 2. click on button "Choose document" and choose a document
# 3. Do step 2 as many times as you want, to uadd as many documents as you like
# 4. After you are done choosing documents, click on the GenerateReport button
# 5. Then you can click on the "open generated report" button, which will automatically
# open up your word document report created from your documents
# 6. When you are done, click "End Program"

# from debug import debug
import logging
# import pdb
import docx
from docx import Document
from docx.shared import RGBColor
from docx.shared import Inches
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from typing import Tuple

from tkinter import scrolledtext
import re
import copy
import time

# This libraries are for opening word document automatically
import os
import platform
import subprocess

# This library is for opening excel document automatically
import xlwings as xw
import pandas as pd
import matplotlib.pyplot as plt

import Gui

from tkinter import messagebox


# Set up the logger for catching errors
```

```
logging.basicConfig(level=logging.ERROR,
                    format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(__name__)

global report1
report1 = Document()
report1.add_heading('All Tags in each document', 0) #create word document
global paragraph0
paragraph0 = report1.add_paragraph()
report1.save('ChildandParentTagsTables.docx')


global report3
report3 = Document()
report3.add_heading('All Tags and Requirement Tracing', 0) #create word document
global paragraph
paragraph = report3.add_paragraph()
report3.save('AllChildandParentTags.docx')

global orphanReport
orphanReport = Document()
orphanReport.add_heading('Orphan Report', 0)
global paragraph2
paragraph2 = orphanReport.add_paragraph()
runnerOrphan = paragraph2.add_run("These are the orphan tags that were found in the
        documents: ")
runnerOrphan.bold = True  # makes it bold
orphanReport.save('orphanReport.docx')

global childlessReport
childlessReport = Document()
childlessReport.add_heading('Childless Report', 0)

global paragraph3
paragraph3 = childlessReport.add_paragraph()
runnerOrphan = paragraph3.add_run("These are the childless tags that were found in the
        documents: ")
runnerOrphan.bold = True  # makes it bold
childlessReport.save('ChildlessTags.docx')

global TBVReport
TBVReport = Document()
TBVReport.add_heading('TBV Tags', 0) #create word document
```

```
global paragraph5
paragraph5 = TBVReport.add_paragraph()
TBVReport.save('TBVReport.docx')

global TBDReport
TBDReport = Document()
TBDReport.add_heading('TBD Tags', 0) #create word document
global paragraph6
paragraph6 = TBDReport.add_paragraph()
TBDReport.save('TBDReport.docx')

global dicts2Copy # This will hold the dicts2 content in all documents
dicts2Copy = {}

global parents2Copy # parents2 list copy
parents2Copy = []

global filtered_L # Will store the ones without a child tag
filtered_L = []

global filtered_LCopy
filtered_LCopy = []

global fullText2Copy
fullText2Copy = []

global parents2 #list of parent tags or child tags
parents2 = []

# creates a dict for parent and child tags
global dicts
dicts = {}

global OrphanChild2
OrphanChild2 = []
global OrphanChild2Copy
orphanChildren2Copy = []

global dicts10
dicts10 = {}
global dicts3
dicts3 = {}  # will hold parentTag and text, Orphan tags
global dicts2
dicts2 = {}  # will hold parentTag and text
```

```
global orphanDicts
orphanDicts = {}  # orphan dictionary

global parents9
parents9 = []

# declaring different lists that will be used to store, tags and sentences
global parentTags
parentTags = []
global parent  # This will be used to store everything
parent = []
global child # Used to Store child tags
child = []
global noChild # Used to Store parentTags with no child
noChild = []
global withChild # Used to Store parentTags with child tag
withChild = []
global parents # Will be used for future function
parents = []

global orphanTagText
orphanTagText = []  # Will be used to hold text of orphanChildTags


# reads the text in the document and use the getcoloredTXT function to get the colored text
def readtxt(filename, color: Tuple[int, int, int]):
    try:
        doc = docx.Document(filename)
        text10 = ""
        fullText = []
        new = []
        global everything
        everything = []  # list of tags and text

        for para in doc.paragraphs:
            # Getting the colored words from the doc
            if (getcoloredTxt(para.runs, color)):
                # Concatenating list of runs between the colored text to single a string
                sentence = "".join(r.text for r in para.runs)
                if len(sentence) > 5:  # this chcekcts if sentence has atleast 5 characters
                    fullText.append(sentence)
                #print(sentence) # Prints everything in the terminal
                everything.append(sentence)
                text10 = sentence
```

```
                parent.append("".join(r.text for r in para.runs))

        #print(fullText)
        global hasChild # Will store the ones with a child tag
        global fullText2 # will store everything found
        global children

        global orphanss
        orphanss = []

        global orphanChildren2 # Will store the orphan child tags for orphanReport
        orphanChildren2 = []
        # Finds the lines without a parentTag
        filtered_L = [value for value in fullText if "[" not in value]

        filtered_L = [s.replace(": ", ":") for s in filtered_L]
        # Finds the lines with a parentTag
        filtered_LCopy.extend(filtered_L)
        hasChild = [value for value in fullText if "[" in value]
        # will store everything found
        fullText2 = [value for value in fullText]
        fullText2 = [s.replace(": ", ":") for s in fullText2]
        fullText2 = [s.replace("[ ", "[") for s in fullText2]
        fullText2 = [s.replace("] ", "]") for s in fullText2]
        fullText2Copy.extend(fullText2)

        return fullText, filtered_L, hasChild, filtered_LCopy, fullText2Copy, fullText2

    except Exception as e:
    # Log an error message
        logging.error('readtxt(): ERROR', exc_info=True)


def getcoloredTxt(runs, color):
    coloredWords, word = [], ""
    try:
        for run in runs:
            if run.font.color.rgb == RGBColor(*color):
                word += str(run.text) # Saves everything found

            elif word != "":  # This will find the parentTags
                coloredWords.append(word)
                parentTags.append(word)
                parents.append(word)
```

```
            word = ""

        if word != "":  # This will find the parentTags
            coloredWords.append(word + "\n")
            # word = removeAfter(word)
            child.append(word)
            withChild.append(word)



    except Exception as e:
        logging.error('getColoredText(): ERROR', e)
    else:
        # Log a success message
        logging.info('getColoredText(): PASS')

    return coloredWords # returns everything found


def generateReport(): #Will generate the report for tags
    try:
        global filepath
        global filepath2
        # create a similar method for opening a folder
        filepath = filedialog.askopenfilename(initialdir="/",
                                              title="",
                                              filetypes = (("text file", "*.txt"),
                                                      ("all files","*.*")))
        file = open(filepath,'r')
        #print(filepath)
        file.close()
        # Will store the filepath to the document as a string
        filepath2 = str(filepath)
        a = (filepath2)
        with open(a) as file_in:
            lines = []
            for line in file_in:
                lines.append(line)
        for line2 in lines:
            print(line2)
            line3 = str(line2)
            line4 = line3.replace('\\', '/')
            line5 = line4.replace("", '')
            line6 = line5.replace("\n", "")
```

```
print(line6)
fullText = readtxt(filename=line6,
            color=(255, 0, 0))
print(line4)

#filtered_L = readtxt(filename=filepath2, #For future use
#                color=(255, 0, 0))
fullText10 = str(fullText)
s = ''.join(fullText10)
w = (s.replace (']', ']\n\n'))
#paragraph = report3.add_paragraph()
paragraph0 = report1.add_paragraph()
#paragraph2 = orphanReport.add_paragraph()
filepath3 = str(line4.rsplit('/', 1)[-1]) # change filepath to something.docx
filepath3 = filepath3.split('.', 1)[0] # removes .docx of the file name
print(filepath3 + " added to the report")
nameOfDoc = (filepath3 + " added to the report\n")
Gui.Txt.insert(tk.END, nameOfDoc) #print in GUI (m = main.py)
#runner = paragraph.add_run("\n" + "Document Name: " + filepath3 + "\n")
#runner.bold = True  # makes the header bold

runner0 = paragraph0.add_run("\n" + "Document Name: " + filepath3 + "\n")
runner0.bold = True  # makes the header bold

# w will be used in the future
w = (w.replace ('([', ''))
w = (w.replace (',', ''))
w = (w.replace (" ", ''))

# creates a table for report 1
#table = report3.add_table(rows=1, cols=2)

# creates a table for report 3
table1 = report1.add_table(rows=1, cols=2)


row1 = table1.rows[0].cells
row1[0].text = 'Front Tag'
row1[1].text = 'Back Tag/tags'

# Adding style to a table
#table.style = 'Colorful List'

table1.style = 'Colorful List'
```

```
    # Now save the document to a location
    #report3.save('report.docx')

    report1.save('ChildandParentTagsTables.docx')

    #orphanReport.save('orphanReport.docx')
    # Adds headers in the 1st row of the table


    e = 0

    child2 = removeAfter(child) #removes everything after the parent tag if there is anything
to remove
    # while loop until all the  parentTags has been added to the report


    parents2 = copy.deepcopy(parentTags) # copy of parent tags list
    parents2Copy.extend(parents2)
    childCopy = copy.deepcopy(child2)
    noParent = []
    noParent2 = []
    global orphanChild
    orphanChild = []
    orphanChildParent = []
    parents9000 = []

    parents2 = [s.replace(" ", "") for s in parents2] # gets rid of space
    while parentTags:
        #row = table.add_row().cells # Adding a row and then adding data in it.
        #row[0].text = parentTags[0] # Adds the parentTag to the table
        #report3.add_paragraph(parentTags[0])
        row1 = table1.add_row().cells # Adding a row and then adding data in it.


        row1[0].text = parentTags[0] # Adds the parentTag to the table


        noParent.append(parentTags[0])


        #for child100 in child2:
         #   report3.add_paragraph(child100)
        for ch in child2:
```

```
            if "[" not in ch:
                child2.remove(ch)



        if e < len(fullText2):  #as long as variable e is not higher than the lines in fullText2
            if fullText2[e] in filtered_LCopy: #filtered_L contains the child tags without a parent
tag
                #report3.add_paragraph(parentTags[0] + " has no child tag")
                orphanChild.append(parentTags[0])
                orphanChildren2.append(parentTags[0])
                #orphanReport.add_paragraph(parentTags[0] + " has no child tag")
                parentTags.remove(parentTags[0]) # Removes that tag after use
                noParent2.append(" ")
                parents9000.append(" ")
                orphanChildParent.append(" ")
                #row[1].text = " " # No parent tag, so adds empty string to that cell
                if child2:
                    if "[" not in child2[0]:
                        row1[1].text = " " # No parent tag, so adds empty string to that cell
                if child2:
                    if "[" not in child2[0]: # if it is not a parent tag
                        child2.remove(child2[0])  # Removed that tag from the list


                #if child2:
                 #   child2.remove(child2[0])  # Removed that tag from the list

                e += 1

            elif fullText2[e] not in filtered_LCopy:
                parentTags.remove(parentTags[0]) # Removes that tag after use
                if child2:
                    #row[1].text = child2[0] #Adds childTag to table

                    if "[" not in child2[0]:
                        child2.remove(child2[0])  # Removed that tag from the list
                        #report3.add_paragraph(child2[0])
                    #report3.add_paragraph(child2[0])
                    row1[1].text = child2[0] #Adds childTag to table
                    parents9000.append(child2[0])
                    noParent.append(child2[0])
                    child2.remove(child2[0])  # Removed that tag from the list
                    e += 1
```

```
  parents9.extend(parents9000)
  orphanChildren2Copy.extend(orphanChildren2)

  # Make sure everything is cleared before the program gets the next document
  child2.clear()
  parentTags.clear()
  child.clear()
  #report3.save('report.docx') #Saves in document "report3"
  orphanReport.save('orphanReport.docx') #Saves in document "orphanReport"
  report1.save('ChildandParentTagsTables.docx') #Saves in document "report3"

  global dicts11
  dicts11 = dict(zip(parents2, childCopy)) #creates a dictionary if there is a child tag and
parent tag
  dicts.update(dicts)

  noParent = [s.replace(" ", "") for s in noParent]
  #dicts3 = dict(zip(noParent, noParent2)) # dictionary for parent tags without child tags
  orphanChild = [s.replace(" ", "") for s in orphanChild]


  #orphanChildren2Copy = copy.deepcopy(orphanChildren2) # copy of orphanChildren2
list

  dicts9000 = dict(zip(orphanChild, orphanChildParent)) # orphan dictionary
  orphanDicts.update(dicts9000)
  OrphanChild2.extend(orphanChild)

  text2 = removeParent(everything) # child tag and text
  # print(text2)
  text3 = removechild(text2)  # only text list
  # print(text3)
  text4 = removeText(text2) # child tags
  # print(text4) #only parent tag list
  #text8 = [s.replace(" ", "") for s in text4]

  parents9000 = [x.strip(' ') for x in parents9000]
  #dicts3 = dict(zip(parents2, childCopy))
  dicts3 = dict(zip(parents2, parents9000))
  dicts10.update(dicts3)
  dicts2 = dict(zip(parents2, text3)) # creates a dictionary with child tags and text
  dicts100 = copy.deepcopy(dicts2)
```

```
            sorted(dicts2.keys()) # sorts the keys in the dictionary
            dicts2Copy.update(dicts100)



            toggle_state2() # This will enable the generate report button
            toggle_state6() # This will enable the open allTags report button
            # for tg in parents2:
            #     if "TBV:" in tg:
            #         TBVReport.add_paragraph(tg)
            # TBVReport.save('TBVReport.docx')

        return filepath2, filtered_L, orphanChild
        return parents2, dicts2, dicts10, dicts2Copy, parents2Copy, fullText2, filtered_LCopy,
         dicts3, orphanDicts, OrphanChild2

    except Exception as e:
        # Log an error message
        logging.error('generateReport(): ERROR', e)
    else:
        # Log a success message
        logging.info('generateReport(): PASS')


def generateReport2():
    try:

        #print("here is dicts10 before:")
        #print(dicts10)
        global dicts11111 # Will be used for the excel report later for child - parent
        dicts11111 = {}
        dicts11111 = copy.deepcopy(dicts10)

        # counters for Excel report2

        global counter1
        counter1 = 2
        global counter2
        counter2 = 1
        global counter3
        counter3 = 0
        global cell
        cell = 0;
        global cell2
```

```
cell2 = 0;



#excelReport2.range("A3").value = 'Childless Tags'

pattern = r'\[([^\]]+)\]'
for key in dicts10:
    if type(dicts10[key]) == str:
        matches2 = re.findall(pattern, (dicts10[key]))
    if len(matches2) > 1:


            #print("There is more than one ']' in the input string.", dicts10[key] )
            dicts10[key] = []
            parents2 = []
            for match in matches2:
                parents2.append(match)

            for tag in parents2:
                #parentChild2.setdefault("[PUMP:SRS:1]",
 ["[PUMP:PRS:0]"]).append("[PUMP:PRS:2]")
                #parentChild2.setdefault(key, [parentChild2[key]]).append(tag)
                #parentChild2[key].append(tag)
                tag = (tag.replace(' ', ''))
                dicts10[key] += [tag]
                #parentChild2.setdefault(key, ["[PUMP:PRS:0]"]).append(tag)
                #parentChild2.setdefault(key, [parentChild2[key]]).append(match)

    else:
        #print("There is only one ']' in the input string.")
        print("")

#print("here is dicts10 after:")
#print(dicts10)



#report3.add_paragraph("all parents:") # header for all parents

parents10 = [] # list of all the parent tag tags
for value11 in dicts10.values():
    # if the value is a list, extend the parents list with the list
    if isinstance(value11, list):
        parents10.extend(value11)
```

```
    # if the value is not a list, append the value to the parents list
    else:
        parents10.append(value11)

# for loop to add the parents to the report
#for parent in parents10:
 #   report3.add_paragraph(parent)


# create a list of all the keys in the dictionary (all child tags)
values_list = list(dicts2Copy.keys())

#  creates a list of all the child tags that are not in the parents list
global childless
childless = []


global TBVTags
TBVTags = []

global TBDTags
TBDTags = []

# for loop to check if the child tag is in the parents list
for element in values_list:
    if "".join(element) not in "".join(parents10):
        childless.append(element)


# sorts the childless list
childless.sort()

# for loop to add the childless tags to the report
for child0 in childless:
    childlessReport.add_paragraph(child0)

childlessReport.save('ChildlessTags.docx') #Saves in document "ChildlessTags.docx"



# declaring counters
m = 0
k = 0
```

```
i = 0
o = 0
z = 0

orphanTagText = removechild(filtered_LCopy)

#dict(sorted(dicts2Copy.items(), key=lambda item: item[1])) # sorts by value/parentTag, not
 working at the moment
#print(parents2Copy)
#print(dicts10)
#print(dicts2Copy)
#print(filtered_LCopy)
#print(fullText2Copy)
#print(parents2Copy)
#print(orphanTagText)
#print(dicts2Copy)
#report3.add_paragraph("\n") # Adds a line space from the table
while m < len(dicts2Copy):
    #print(m)
    #if fullText2Copy[k] not in filtered_LCopy:
    if z < len(dicts2Copy) and dicts2Copy:
        z += 1
        duplicates = []
        for key, value in dicts2Copy.items():

            m += 1
            if k < len(fullText2Copy) and fullText2Copy[k] not in filtered_LCopy:
                #for key, value in dicts2Copy.items() and key, value in dicts3.items(): #work on
 this here and try
                #report3.add_paragraph("\n")
                stringKey = str(key)
                stringKey2 = (stringKey.replace(' ', ''))
                if str(stringKey2) in dicts10: # if the key is in the dictionary
                    text = dicts10[str(stringKey2)]



                if isinstance(text, list):
                    #print("it is a list")
                    #report3.add_paragraph("List tags found") # display the parent tag, included
 brackets

                    for tag in text:
                        #tag = ("[" + tag)
```

```
PTags = tag.split(']')
PTags = [s.strip() + ']' for s in PTags]
tag.strip()

#report3.add_paragraph(tag)
if (str(tag) in duplicates):
    #print("in duplicates")
    print("")



else:
    parentTag1 = ('['+tag+']')

    #cell = str('A'+ str(counter1))
    #cell2 = str(str(parentTag1))
    #xcelReport2.range(cell).value = cell2
    counter1 += 2 # counter for excel report
    counter2 += 1 # counter for excel report

    #wb2.save('AllTags.xlsx') # Saving excel report as 'AllTags.xlsx'
    if "TBV:" in parentTag1:
        TBVReport.add_paragraph(parentTag1)
        TBVReport.save('TBVReport.docx')
        TBVTags.append(parentTag1)
    if "TBD:" in parentTag1:
        TBDReport.add_paragraph(parentTag1)
        TBDReport.save('TBDReport.docx')
        TBDTags.append(parentTag1)

    report3.add_paragraph(parentTag1)
    tag.strip()
    duplicates.append(str(tag))



    for x in PTags:
        #report3.add_paragraph(x)

        keyCheck = (x.replace('[', ''))
        keyCheck2 = (keyCheck.replace(']', ''))
        keyCheck3 = (keyCheck2.replace(']', ''))
        keyCheck4 = (keyCheck3.replace(' ', ''))
```

```
                    keyCheck4.split()



                    if keyCheck4 in dicts2Copy:  # Checks if text of parent tag is found
                        if dicts2Copy[str(keyCheck4)] != "" and
dicts2Copy[str(keyCheck4)] != " ":
                            if "TBV:" in parentTag1:
                                TBVReport.add_paragraph(dicts2Copy[str(keyCheck4)])
                                TBVReport.save('TBVReport.docx')
                            if "TBD:" in parentTag1:
                                TBDReport.add_paragraph(dicts2Copy[str(keyCheck4)])
                                TBDReport.save('TBDReport.docx')


                            report3.add_paragraph(dicts2Copy[str(keyCheck4)])



                        #orphanReport.add_paragraph(dicts2Copy[str(keyCheck4)])

                    else:
                        report3.add_paragraph("Requirement text not found")
                        orphanChildren2Copy.append(str(keyCheck4))
                        #orphanReport.add_paragraph("Requirement text not found")
                    #print(dicts10[str(key)])
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    #for dicts10[str(stringKey)] in dicts10:
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    for b in PTags:
                        b = (b.replace(']', ''))
                        #report3.add_paragraph("I'm b")
                        #report3.add_paragraph(b)
                        #report3.add_paragraph("I'm tag")
                        #report3.add_paragraph(tag)


                        if b == tag:
                            i += 1
                            hx = tag
                            #report3.add_paragraph("I'm here")

                            keys = [h for h, v in dicts10.items() if check_string(hx, v)] # finds
all the child tags
```

```
                                #report3.add_paragraph("I'm keys")
                                #report3.add_paragraph(keys)
                                k += 1
                                for item in keys: #keys are child tags of hx/the parent tag

                                    if item != "" and item!= " ":
                                        report3.add_paragraph(item, style='List Bullet')
                                        para = report3.add_paragraph(dicts2Copy[str(item)])
                                        para.paragraph_format.left_indent = Inches(0.25) # adds
indentation of text
                                        if "TBD:" in parentTag1:
                                            TBDReport.add_paragraph(item, style='List Bullet')
                                            para3 = report3.add_paragraph(dicts2Copy[str(item)])
                                            para3.paragraph_format.left_indent = Inches(0.25) # adds
indentation of text
                                            TBDReport.save('TBDReport.docx')

                                        if "TBV:" in parentTag1:
                                            TBVReport.add_paragraph(item, style='List Bullet')
                                            if str(item) in dicts2Copy:
                                                para2 =
TBVReport.add_paragraph(dicts2Copy[str(item)])
                                                para2.paragraph_format.left_indent = Inches(0.25) #
adds indentation of text
                                            #TBVReport.add_paragraph("here1")
                                            TBVReport.save('TBVReport.docx')
                                            stringKey = str(item)
                                            stringKey2 = (stringKey.replace(' ', ''))
                                            duplicates2 = []
                                            if str(stringKey2) in dicts10: # if the key is in the dictionary
                                                #for key in dicts10:
                                                    if item in dicts10:
                                                        hx = item
                                                                #report3.add_paragraph("I'm here")

                                                        keys = [h for h, v in dicts10.items() if
check_string(hx, v)] # finds all the child tags
                                                                #report3.add_paragraph("I'm keys")
                                                                #report3.add_paragraph(keys)
                                                        k += 1
                                                        for item1 in keys: #keys are child tags of hx/the
parent tag

                                                            if item1 != "" and item1!= " ":
```

```
                                                    if item1 not in duplicates2:
                                                        duplicates2.append(item1)
                                                        tag2 = TBVReport.add_paragraph(item1)
                                                        tag2.paragraph_format.left_indent =
Inches(0.50) # adds indentation of text
                                                        para =
TBVReport.add_paragraph(dicts2Copy[str(item1)])
                                                        para.paragraph_format.left_indent =
Inches(0.50) # adds indentation of text
                                                        TBVReport.save('TBVReport.docx')


                                            TBVReport.save('TBVReport.docx')



                                        counter2 = counter1 - 1
                                        cell = str('B'+ str(counter2))
                                        cell2 = str(item)
                                        #excelReport2.range(cell).value = cell2
                                        counter2 += 1
                                        counter1 += 1


                                        #wb2.save('AllTags.xlsx') # Saving excel report as
'AllTags.xlsx'

                                    #wb2.save('AllTags.xlsx') # Saving excel report as 'AllTags.xlsx'
                                    report3.add_paragraph("\n")
                                    counter2 += 1
                                    counter1 += 1
                                    #excelReport2.autofit()



                        else:
                            #print("not a list")
                            PTags = text.split(']')
                            PTags = [s.strip() + ']' for s in PTags]
                            PTags.pop()
                            hx10 = text
                            hx10 = hx10.replace('[', '')
                            hx10 = hx10.replace(']', '')
                            #tag.strip()
```

```
                if (str(hx10) in duplicates):
                    #print("in duplicates")
                    print("")


                else:
                    #parentTag1 = ('['+tag+']')
                    #report3.add_paragraph(parentTag1)
                    #tag.strip()


                    for x in PTags:

                        #report3.add_paragraph(str(text))
                        keyCheck = (x.replace('[', ''))
                        keyCheck2 = (keyCheck.replace(']', ''))
                        keyCheck3 = (keyCheck2.replace(']', ''))
                        keyCheck4 = (keyCheck3.replace(' ', ''))
                        report3.add_paragraph(x) # display the parent tag, included brackets
                        if "TBV:" in x:
                                        TBVReport.add_paragraph(x)
                                        TBVReport.save('TBVReport.docx')
                                        TBVTags.append(x)
                        if "TBD:" in x:
                                        TBDReport.add_paragraph(item)
                                        TBDReport.save('TBDReport.docx')
                                        TBDTags.append(x)

                        cell = str('A'+ str(counter1))
                        cell2 = str(str(x))
                        #excelReport2.range(cell).value = cell2
                        counter1 += 2 # counter for excel report
                        counter2 += 1 # counter for excel report

                        #wb2.save('AllTags.xlsx') # Saving excel report as 'AllTags.xlsx'

                        if keyCheck4 in dicts2Copy:  # Checks if text of parent tag is found
                            if dicts2Copy[str(keyCheck4)] != "" and dicts2Copy[str(keyCheck4)] != "
            ":
                                report3.add_paragraph(dicts2Copy[str(keyCheck4)])
                                if "TBD:" in keyCheck4:
                                        TBDReport.add_paragraph(dicts2Copy[str(keyCheck4)])
                                        TBDReport.save('TBDReport.docx')
                                if "TBV:" in keyCheck4:
```

```
                                TBVReport.add_paragraph(dicts2Copy[str(keyCheck4)])
                                TBVReport.save('TBVReport.docx')
                    #orphanReport.add_paragraph(dicts2Copy[str(keyCheck4)])

                else:
                    report3.add_paragraph("Requirement text not found")
                    orphanChildren2Copy.append(str(keyCheck4))
                    #orphanReport.add_paragraph("Requirement text not found")
                #print(dicts10[str(key)])
                #report3.add_paragraph(dicts10[str(stringKey)])
                #for dicts10[str(stringKey)] in dicts10:
                #report3.add_paragraph(dicts10[str(stringKey)])
                for b in PTags:


                    if b == dicts10[str(stringKey2)]:
                        i += 1
                        text.strip()

                        hx = text
                        hx = hx.replace('[', '')
                        hx = hx.replace(']', '')

                        duplicates.append(str(hx))
                        #report3.add_paragraph(str(hx))

                        keys = [h for h, v in dicts10.items() if check_string(hx, v)]
                        # finds all the child tags
                        #print(keys)
                        k += 1

                        #if keys:
                        #   print("list is not empty")
                        #else:
                        #    report3.add_paragraph("It is an orphan tag")

                        for item in keys: #keys are child tags of hx/the parent tag

                            if item != "" and item!= " ":
                                report3.add_paragraph(item, style='List Bullet')
                                para = report3.add_paragraph(dicts2Copy[str(item)])
                                para.paragraph_format.left_indent = Inches(0.25) # adds
indentation of text
                                if "TBD:" in x:
```

```
                                    TBDReport.add_paragraph(item, style='List Bullet')
                                    para = TBDReport.add_paragraph(dicts2Copy[str(item)])
                                    para.paragraph_format.left_indent = Inches(0.25) # adds
indentation of text
                                    TBDReport.save('TBDReport.docx')
                                if "TBV:" in x:
                                    TBVReport.add_paragraph(item, style='List Bullet')
                                    if str(item) in dicts2Copy:
                                        para2 =
TBVReport.add_paragraph(dicts2Copy[str(item)])
                                        para2.paragraph_format.left_indent = Inches(0.25) #
adds indentation of text
                                    #TBVReport.add_paragraph("here1")
                                    TBVReport.save('TBVReport.docx')
                                    stringKey = str(item)
                                    stringKey2 = (stringKey.replace(' ', ''))
                                    duplicates2 = []
                                    if str(stringKey2) in dicts10: # if the key is in the dictionary
                                        #for key in dicts10:
                                            if item in dicts10:
                                                hx = item
                                                        #report3.add_paragraph("I'm here")

                                                keys = [h for h, v in dicts10.items() if
check_string(hx, v)] # finds all the child tags
                                                        #report3.add_paragraph("I'm keys")
                                                        #report3.add_paragraph(keys)
                                                k += 1
                                                for item1 in keys: #keys are child tags of hx/the
parent tag

                                                    if item1 != "" and item1!= " ":
                                                        if item1 not in duplicates2:
                                                            duplicates2.append(item1)
                                                            tag2 = TBVReport.add_paragraph(item1)
                                                            tag2.paragraph_format.left_indent =
Inches(0.50) # adds indentation of text
                                                            para =
TBVReport.add_paragraph(dicts2Copy[str(item1)])
                                                            para.paragraph_format.left_indent =
Inches(0.50) # adds indentation of text
                                                            TBVReport.save('TBVReport.docx')
```

```
                                        TBVReport.save('TBVReport.docx')


                                    counter2 = counter1 - 1
                                    cell = str('B'+ str(counter2))
                                    cell2 = str(item)
                                    #excelReport2.range(cell).value = cell2
                                    counter2 += 1
                                    counter1 += 1

                                report3.add_paragraph("\n")
                                counter2 += 1
                                counter1 += 1
                                #excelReport2.autofit()

                    #report3.add_paragraph("\n") # Adds a line space
                    #print(k)
                    #print(m)
                    #report3.add_paragraph(key, style='List Bullet')
                    #para = report3.add_paragraph(value)
                    #para.paragraph_format.left_indent = Inches(0.25) # adds indentation ot text
                    #stringKey = dicts2Copy[str(key)]
                    #stringKey2 = (stringKey.replace(' ', ''))
                #if k < len(fullText2Copy):
            #elif k < len(fullText2Copy) and fullText2Copy[k] in filtered_LCopy:
            elif k < len(fullText2Copy) and fullText2Copy[k] in filtered_LCopy:
                k += 1
                #report3.add_paragraph("\n")
                if i < len(parents2Copy):
                    #report3.add_paragraph(parents2Copy[i])
                    #orphanReport.add_paragraph(parents2Copy[i])
                    #paragraph2.add("\n" +parents2Copy[i] + " Is an orphanTag" + "\n")

                    #if "TBV:" in parentTag1:
                    #    TBVReport.add_paragraph(item, style='List Bullet')
                    #    TBVReport.save('TBVReport.docx')
                    # if "TBV:" in parentTag1:
                    #    TBVReport.add_paragraph(item, style='List Bullet')



                    TBVReport.save('TBVReport.docx')
                    #print(parents2Copy[i])
                    #print(orphanTagText[o])
```

```
                print("")
                #orphanss.append(parents2Copy[i])
            if o < len(orphanTagText):
                #report3.add_paragraph(orphanTagText[o])
                #orphanReport.add_paragraph(orphanTagText[o])
                print("")
            o += 1
            if i < len(parents2Copy):
                print("")
                #orphanReport.add_paragraph(parents2Copy[i] + " is an orphan tag")
            #m += 1

            i += 1
            #del dicts2Copy[list(dicts2Copy.keys())[0]] # deletes the first item in dicts2Copy


    # Description, this function is removing some linees, but not all, maybe need more work
    # We check if the length of the paragraph text is less than 4 using the len() function,
    # and if so, we remove the paragraph using the _element.clear() method. Finally
    # iterate over all the paragraphs in the document
    #for i in range(len(report3.paragraphs)):
        # check if the paragraph contains less than 4 characters or a string of length less than 4
     #   if len(report3.paragraphs[i].text) < 4:
            # remove the paragraph
      #      report3.paragraphs[i]._element.clear()
            # save the modified document
       #     report3.save('AllChildandParentTags.docx')

    #if parents2Copy:
    #    for tg in parents2Copy:
    #        if "TBV:" in tg:
    #            TBVReport.add_paragraph(tg)
    #            TBVReport.save('TBVReport.docx')

    #wb2.save('AllTags.xlsx')
    msg1 = ("\nYou can now open up your Word and Excel reports\n")
    Gui.Txt.insert(tk.END, msg1) #print in GUI
    msg2 = ("\nDifferent reports include:\n")
    Gui.Txt.insert(tk.END, msg2) #print in GUI
    msg3 = ("a. All of the Tags found organized in tables\n")
    Gui.Txt.insert(tk.END, msg3) #print in GUI
    msg4 = ("b. All Child and Parent Tags\n")
    Gui.Txt.insert(tk.END, msg4) #print in GUI
    msg5 = ("c. Orphan Tags\n")
```

```
        Gui.Txt.insert(tk.END, msg5) #print in GUI
        msg6 = ("d. Childless Tags\n")
        Gui.Txt.insert(tk.END, msg6) #print in GUI
        msg7 = ("e. TBV Tags\n")
        Gui.Txt.insert(tk.END, msg7) #print in GUI
        msg8 = ("f. TBD Tags\n")
        Gui.Txt.insert(tk.END, msg8) #print in GUI
        msg9 = ("g. Tags and Requirements Excel report\n")
        Gui.Txt.insert(tk.END, msg9) #print in GUI
        msg10 = ("h. Relationship Trees Excel Report\n")
        Gui.Txt.insert(tk.END, msg10) #print in GUI
        report3.save('AllChildandParentTags.docx')
        toggle_state() #This will enable the getDoc button
        Gui.genRep.config(state="disabled") # This will disable the generate report button
        TBVReport.save('TBVReport.docx') # saves the TBV report
        toggle_state3() # this will re-enable excel report button
        #toggle_state5() # This will enable the generate orphan report button
        toggle_state7() #This will enable the getChildless document button
        toggle_state9() # This will enable the get TBV button
        toggle_state10() # This will enable the get TBD button
        toggle_state8() # This will enable the getExcel2 report button
        toggle_state13() # This will enable the guiTree View button
        orphanGenReport()



    except Exception as e:
        # Log an error message
        logging.error('generateReport2(): ERROR', e)
    else:
        # Log a success message
        logging.info('generateReport2(): PASS')

    """
        elif not dicts2Copy: # this is for orphan tags
            dict3 = dict(dicts2.items() - dicts3.items())
            for key, value in dicts3.items():
                report3.add_paragraph("\n")
                report3.add_paragraph(key)
                report3.add_paragraph(value)
                report3.add_paragraph(key + " is an orphan tags")
                m += 1
    """
```

```
def orphanGenReport():
    duplicates = []
    try:
        """
        # declaring counters
        m = 0
        k = 0
        i = 0
        o = 0
        z = 0

        orphanTagText = removechild(filtered_LCopy)
        while m < len(dicts2Copy):
            #print(m)
            #if fullText2Copy[k] not in filtered_LCopy:
            if z < len(dicts2Copy) and dicts2Copy:
                z += 1

                for key, value in dicts2Copy.items():
                    #orphanReport.add_paragraph("\n")
                    m += 1
                    if k < len(fullText2Copy) and fullText2Copy[k] not in filtered_LCopy:
                        #for key, value in dicts2Copy.items() and key, value in dicts3.items(): #work on
        this here and try
                        #report3.add_paragraph("\n")
                        stringKey = str(key)
                        stringKey2 = (stringKey.replace(' ', ''))
                        if stringKey2 in dicts10:
                            text = dicts10[str(stringKey2)]


                        if isinstance(text, list):
                            #print("it is a list")
                            print("")
                            #report3.add_paragraph("List tags found") # display the parent tag, included
        brackets

                            for tag in text:
                                #tag = ("[" + tag)
                                PTags = tag.split(']')
                                PTags = [s.strip() + ']' for s in PTags]
                                tag.strip()
                                if (str(tag) in duplicates):
                                    #print("in duplicates")
```

```
                print("")


            else:

                #report3.add_paragraph(tag)
                #tag.strip()
                duplicates.append(str(tag))


                for x in PTags:
                    #report3.add_paragraph(x)

                    keyCheck = (x.replace('[', ''))
                    keyCheck2 = (keyCheck.replace(']', ''))
                    keyCheck3 = (keyCheck2.replace(']', ''))
                    keyCheck4 = (keyCheck3.replace(' ', ''))
                    keyCheck4.split()



                    if keyCheck4 in dicts2Copy:  # Checks if text of parent tag is found
                        if dicts2Copy[str(keyCheck4)] != "" and
dicts2Copy[str(keyCheck4)] != " ":
                            #report3.add_paragraph(dicts2Copy[str(keyCheck4)])
                            print("")
                        #orphanReport.add_paragraph(dicts2Copy[str(keyCheck4)])

                    else:
                        print("")
                        #orphanChildren2Copy.append(str(keyCheck4))
                        #report3.add_paragraph("Requirement text not found")
                        #orphanReport.add_paragraph("Requirement text not found")
                    #print(dicts10[str(key)])
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    #for dicts10[str(stringKey)] in dicts10:
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    for b in PTags:
                        b = (b.replace(']', ''))
                        #report3.add_paragraph("I'm b")
                        #report3.add_paragraph(b)
                        #report3.add_paragraph("I'm tag")
                        #report3.add_paragraph(tag)
```

```
                        if b == tag:
                            i += 1
                            hx = tag
                            #report3.add_paragraph("I'm here")
                            keys = [h for h, v in dicts10.items() if check_string(hx, v)] # finds
    all the child tags
                            #report3.add_paragraph("I'm keys")
                            #report3.add_paragraph(keys)
                            k += 1
                            for item in keys: #keys are child tags of hx/the parent tag

                                if item != "" and item!= " ":
                                    print("")
                                    #report3.add_paragraph(item, style='List Bullet')
                                    #para = report3.add_paragraph(dicts2Copy[str(item)])
                                    #para.paragraph_format.left_indent = Inches(0.25) # adds
    indentation of text




            else:
                print("")
                PTags = text.split(']')
                PTags = [s.strip() + ']' for s in PTags]
                PTags.pop()

                for x in PTags:
                    keyCheck = (x.replace('[', ''))
                    keyCheck2 = (keyCheck.replace(']', ''))
                    keyCheck3 = (keyCheck2.replace(']', ''))
                    keyCheck4 = (keyCheck3.replace(' ', ''))
                    #report3.add_paragraph(x) # display the parent tag, included brackets

                    if keyCheck4 in dicts2Copy:  # Checks if text of parent tag is found
                        if dicts2Copy[str(keyCheck4)] != "" and dicts2Copy[str(keyCheck4)] != " ":
                            print("")
                            #report3.add_paragraph(dicts2Copy[str(keyCheck4)])
                        #orphanReport.add_paragraph(dicts2Copy[str(keyCheck4)])

                    else:
                        #report3.add_paragraph("Requirement text not found")
```

```
                        #orphanChildren2Copy.append(str(keyCheck4))
                        print("")
                        #orphanReport.add_paragraph("Requirement text not found")
                    #print(dicts10[str(key)])
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    #for dicts10[str(stringKey)] in dicts10:
                    #report3.add_paragraph(dicts10[str(stringKey)])
                    for b in PTags:



                        if b == dicts10[str(stringKey2)]:
                            i += 1
                            hx = dicts10[str(stringKey2)]
                            keys = [h for h, v in dicts10.items() if v == hx] # finds all the child tags
                            #print(keys)
                            k += 1
                            for item in keys: #keys are child tags of hx/the parent tag

                                if item != "" and item!= " ":
                                    print("")
                                    #report3.add_paragraph(item, style='List Bullet')
                                    #para = report3.add_paragraph(dicts2Copy[str(item)])
                                    #para.paragraph_format.left_indent = Inches(0.25) # adds
indentation of text

                    #report3.add_paragraph("\n") # Adds a line space
                    #print(k)
                    #print(m)
                    #report3.add_paragraph(key, style='List Bullet')
                    #para = report3.add_paragraph(value)
                    #para.paragraph_format.left_indent = Inches(0.25) # adds indentation ot text
                    #stringKey = dicts2Copy[str(key)]
                    #stringKey2 = (stringKey.replace(' ', ''))
                #if k < len(fullText2Copy):
            #elif k < len(fullText2Copy) and fullText2Copy[k] in filtered_LCopy:


            elif k < len(fullText2Copy) and fullText2Copy[k] in filtered_LCopy:
                k += 1
                #orphanReport.add_paragraph("\n")
                if i < len(parents2Copy):
                    orphanss.append(parents2Copy[i]) # adds orphan tags to a list
                    #orphanReport.add_paragraph(parents2Copy[i])
```

```
                    #orphanReport.add_paragraph(parents2Copy[i] + " is an orphan tag")
                    #print(parents2Copy[i])
                    #print(orphanTagText[o])
                    print("")
                if o < len(orphanTagText):
                    #orphanReport.add_paragraph(orphanTagText[o])
                    print("")

                o += 1
                if i < len(parents2Copy):
                    print("")
                    #orphanReport.add_paragraph(parents2Copy[i] + " is an orphan tag")
                i += 1

        #orphanss.sort() # sorts the list of orphan tags
        #orphanChildren2Copy.sort() # sorts the list of orphan child tags
        """

        #for orph in orphanss:
        #    orphanReport.add_paragraph(orph)
        #orphanReport.add_paragraph("Orphan Tags2: ")


        for orph5 in orphanChildren2Copy:
            orphanReport.add_paragraph(orph5)
            print(orph5)

        orphanReport.save('orphanReport.docx')
        toggle_state4() # This will enable the open orphan report button
        return dicts2Copy

    except Exception as e:
        # Log an error message
        logging.error('orphanReport(): ERROR', e)
    else:
        # Log a success message
        logging.info('orphanReport(): PASS')



def removeParent(text): #removes parent tags or child tags
    try:
        childAfter = []
        for line in text:
```

```
            childAfter = [i.rsplit('[', 1)[0] for i in text] # removes parent tags
            childAfter = [re.sub("[\(\[].*?[\)\]]", "", e) for e in childAfter]  # removes parent tags that are
         left
            childAfter = [re.sub("[\{\[].*?[\)\}]", "", e) for e in childAfter]  # removes "pass", "fail", etc.
        return childAfter

    except Exception as e:
        # Log an error message
        logging.error('removeParent(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('removeParent(): PASS')


def removeText(text6): #this should remove everything before the parent tag
    try:
        childAfter = [s.split(None, 1)[0] if len(s.split(None, 1)) >= 2 else '' for s in text6]
        return childAfter
    except Exception as e:
        # Log an error message
        logging.error('removeText(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('removeText(): PASS')


def removeAfter(childtags): #removes everything after the  tag, example "pass"
    try:
        seperator = ']'
        childAfter = [i.rsplit(']', 1)[0] + seperator for i in childtags]

        return childAfter
    except Exception as e:
        # Log an error message
        logging.error('removeAfter(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('removeAfter(): PASS')


def removechild(text): #removes child, this one needs fixing
    try:
        mylst = []
        mylst = [s.split(None, 1)[1] if len(s.split(None, 1)) >= 2 else '' for s in text]
```

```
        return mylst
    except Exception as e:
        # Log an error message
        logging.error('removechild(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('removechild(): PASS')

# This function will open up the report automatically
def getDocumentTable():
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'ChildandParentTagsTables.docx'])
        elif platform.system() == 'Windows':
            os.startfile('ChildandParentTagsTables.docx')
        # os.startfile(report3) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', report1)
    except Exception as e:
        # Log an error message
        logging.error('getDocumentTable(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('getDocumentTable(): PASS')


# This function will open up the report automatically
def getDocument():
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'AllChildandParentTags.docx'])
        elif platform.system() == 'Windows':
            os.startfile('AllChildandParentTags.docx')
        # os.startfile(report3) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', report3)
    except Exception as e:
        # Log an error message
        logging.error('getDocument(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('getDocument(): PASS')

def getOrphanDocument():
```

```
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'orphanReport.docx'])
        elif platform.system() == 'Windows':
            os.startfile('orphanReport.docx')
        # os.startfile(orphanReport) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', orphanGenReport)
    except Exception as e:
        # Log an error message
        logging.error('getOrphanDocument(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('getOrphanDocument(): PASS')


def getChildlessDocument():
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'ChildlessTags.docx'])
        elif platform.system() == 'Windows':
            os.startfile('ChildlessTags.docx')
        # os.startfile(orphanReport) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', generateReport2)
    except Exception as e:
        # Log an error message
        logging.error('getChildlessDocument(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('getChildlessDocument(): PASS')

def getTBV():
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'TBVReport.docx'])
        elif platform.system() == 'Windows':
            os.startfile('TBVReport.docx')
        # os.startfile(tbvReport) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', generateReport2)
    except Exception as e:
        # Log an error message
        logging.error('getTBV(): ERROR', exc_info=True)
```

```
    else:
        # Log a success message
        logging.info('getTBV(): PASS')

def getTBD():
    try:
        if platform.system() == 'Darwin':
            subprocess.check_call(['open', 'TBDReport.docx'])
        elif platform.system() == 'Windows':
            os.startfile('TBDReport.docx')
        # os.startfile(tbvReport) # try either one for windows if the first option gives error
        else:
            subprocess.call('xdg-open', generateReport2)
    except Exception as e:
        # Log an error message
        logging.error('getTBD(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('getTBD(): PASS')

# Creates an excel report
def createExcel():

        wb = xw.Book()
        excelReport = wb.sheets[0]
        excelReport.name = "Report"

        # creating a Dataframe object from a list
        # of tuples of key, value pair
        df = pd.DataFrame(list(dicts2Copy.items()))
        # Dictionary For child and parent tag
        df2 = pd.DataFrame(list(dicts11111.items()))

        # For Orphan Tags
        df3 = pd.DataFrame(orphanChildren2Copy)

        # For Childless Tags
        df4 = pd.DataFrame(childless)

        df5 = pd.DataFrame(TBVTags)

        df6 = pd.DataFrame(TBDTags)
```

```
# For childTag -Text
excelReport.range("A1").value = df

# Select the range with the dataframe
#data_range = ws.range('A1').expand()
# Drop the indexes
#data_range.options(index=False).value

# Listing out the Orphan Tags
excelReport.range("G1").value = df3
df3 = df.reset_index(drop=True)

# Listing out the Childless Tags
excelReport.range("I1").value = df4
df4 = df.reset_index(drop=True)

# Listing out the TBV Tags
excelReport.range("K1").value = df5
df5 = df.reset_index(drop=True)

# Listing out the TBD Tags
excelReport.range("M1").value = df6
df6 = df.reset_index(drop=True)



# Adding childTag header
excelReport.range("B1").value = 'Child Tag'
excelReport.range("B1").font.Size = 14 # Change font size
excelReport.range("B1").font.ColorIndex = 2 # Change font color
excelReport.range('B1:B1').color = (255, 0, 0) # Change cell background color

# Adding Text header
excelReport.range("C1").value = 'Text'
excelReport.range("C1").font.Size = 14 # Change font size
excelReport.range("C1").font.ColorIndex = 2 # Change font color
excelReport.range('C1:C1').color = (0,255,0) # Change cell background color

# For the childTag - parentTag
excelReport.range("D1").value = df2

excelReport.range("E1").value = "Child Tag"
excelReport.range("E1").font.Size = 14
excelReport.range("E1").font.ColorIndex = 2
```

```
        excelReport.range("E1:E1").color = (255, 0, 0)

        # Adding parentTag header
        excelReport.range("F1").value = 'Parent Tag'
        excelReport.range("F1").font.Size = 14 # Change font size
        excelReport.range("F1").font.ColorIndex = 2 # Change font color
        excelReport.range('F1:F1').color = (128, 128, 128) # Change cell background color

        # Adding OrphanTags header
        excelReport.range("H1").value = 'Orphan Tags'
        excelReport.range("H1").font.Size = 14 # Change font size
        excelReport.range("H1").font.ColorIndex = 2 # Change font color
        excelReport.range('H1:H1').color = (255, 128, 0) # Change cell background color

        # Adding ChildlessTags header
        excelReport.range("J1").value = 'Childless Tags'
        excelReport.range("J1").font.Size = 14 # Change font size
        excelReport.range("J1").font.ColorIndex = 2 # Change font color
        excelReport.range('J1:J1').color = (150, 75, 0) # Change cell background color

        # Adding TBVTags header
        excelReport.range("L1").value = 'TBV Tags'
        excelReport.range("L1").font.Size = 14 # Change font size
        excelReport.range("L1").font.ColorIndex = 2 # Change font color
        excelReport.range('L1:L1').color = (150, 75, 0) # Change cell background color

        # Adding TBDTags header
        excelReport.range("N1").value = 'TBD Tags'
        excelReport.range("N1").font.Size = 14 # Change font size
        excelReport.range("N1").font.ColorIndex = 2 # Change font color
        excelReport.range('N1:N1').color = (150, 75, 0) # Change cell background color


        excelReport.autofit()

        for key in dicts2:
            wb.sheets[0].append([key, dicts2[key]])
        wb.save('Tags&Requirements.xlsx') # Saving excel report as 'Tags&Requirements.xlsx'

"""
# Creates an excel report
def createExcel2():
    try:
        # book_arr = xw.App().books.add()
```

```
# wb = book_arr.add()
# wb.title = "Report"



# excelReport = wb.sheets.add("Report")

#excelReport.name = report
excelReport2.range("B1").value = "Children"
excelReport2.range("B1").font.Size = 18 # Change font size
excelReport2.range("B1").font.ColorIndex = 2 # Change font color
excelReport2.range('B1:B1').color = (255, 0, 0) # Change cell background color

excelReport2.range("A1").value = 'Parents'
excelReport2.range("A1").font.Size = 18 # Change font size
excelReport2.range("A1").font.ColorIndex = 2 # Change font color
excelReport2.range('A1:A1').color = (0, 0, 255) # Change cell background color



# For Orphan Tags
df3 = pd.DataFrame(orphanChildren2Copy)

# For Childless Tags
df4 = pd.DataFrame(childless)

# For TBV Tags
df5 = pd.DataFrame(TBVTags)

# For TBD Tags
df6 = pd.DataFrame(TBDTags)



# Select the range with the dataframe
#data_range = ws.range('A1').expand()
# Drop the indexes
#data_range.options(index=False).value

# Listing out the Orphan Tags
excelReport2.range("H1").value = df3
df3 = df3.reset_index(drop=True)
```

```
# Listing out the Childless Tags
excelReport2.range("K1").value = df4
df4 = df4.reset_index(drop=True)

# Listing out the TBV Tags
excelReport2.range("N1").value = df5
df5 = df5.reset_index(drop=True)

# Listing out the TBD Tags
excelReport2.range("Q1").value = df6
df6 = df6.reset_index(drop=True)



# Adding OrphanTags header
excelReport2.range("I1").value = 'Orphan Tags'
excelReport2.range("I1").font.Size = 14 # Change font size
excelReport2.range("I1").font.ColorIndex = 2 # Change font color
excelReport2.range('I1:I1').color = (255, 128, 0) # Change cell background color

# Adding ChildlessTags header
excelReport2.range("L1").value = 'Childless Tags'
excelReport2.range("L1").font.Size = 14 # Change font size
excelReport2.range("L1").font.ColorIndex = 2 # Change font color
excelReport2.range('L1:L1').color = (150, 75, 0) # Change cell background color

# Adding TBVTags header
excelReport2.range("O1").value = 'TBV Tags'
excelReport2.range("O1").font.Size = 14 # Change font size
excelReport2.range("O1").font.ColorIndex = 2 # Change font color
excelReport2.range('O1:O1').color = (150, 75, 0) # Change cell background color

# Adding TBDTags header
excelReport2.range("R1").value = 'TBD Tags'
excelReport2.range("R1").font.Size = 14 # Change font size
excelReport2.range("R1").font.ColorIndex = 2 # Change font color
excelReport2.range('R1:R1').color = (150, 75, 0) # Change cell background color



excelReport2.autofit()
```

```
        wb2.save('AllTags.xlsx') # Saving excel report as 'AllTags.xlsx'
    except Exception as e:
        # Log an error message
        logging.error('createExcel2(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('createExcel2(): PASS')
"""

def toggle_state(): # this will re-enable getDoc button
    Gui.getDoc.config(state="normal")

def toggle_state2(): # this will re-enable generate report button
    Gui.genRep.config(state="normal")

def toggle_state3(): # this will re-enable excel report button
    Gui.getExcel.config(state="normal")

def toggle_state4(): # this will re-enable word report button for orphan tags
    Gui.getOrphanDoc.config(state="normal")

#def toggle_state5(): # this will re-enable excel report button for orphan tags
#    Gui.getOrphan.config(state="normal")

def toggle_state6(): # this will re-enable allTags report button for tables
    Gui.allTagsButton.config(state="normal")

def toggle_state7(): # this will re-enable childless report button
    Gui.getChildlessDoc.config(state="normal")

def toggle_state8(): # this will re-enable excelreport 2 button
    Gui.getExcel2.config(state="normal")

def toggle_state9(): # this will re-enable tbv report button
    Gui.getTBVdoc.config(state="normal")

def toggle_state10(): # this will renable tbd report button
    Gui.getTBDdoc.config(state="normal")

def toggle_state13(): # this will renable treeview button
    Gui.TreeDiagram.config(state="normal")

#def check_string(string1, string2): # checks if a string1 is in string2
```

```
#   if isinstance(string2, str):
 #      string2 = [string2]
  # pattern = r'{}(?!\d)'.format(re.escape(string1))
   #for s in string2:
    #   match = re.search(pattern, s)
     #  if match is not None:
      #     return True
   #return False



def check_string2(string1, string2):
    if not isinstance(string1, str):
        return False
    if isinstance(string2, str):
        string2 = [string2]
    elif not isinstance(string2, list):
        return False

    pattern = r'{}(?!\d)'.format(re.escape(string1))

    for s in string2:
        if not isinstance(s, str):
            continue
        match = re.search(pattern, s)
        if match is not None:
            return True

    return False


def check_string(string1, string2):
    if not isinstance(string1, str):
        return False
    if isinstance(string2, str):
        string2 = [string2]
    elif not isinstance(string2, list):
        return False

    pattern = r'(?<=\[)?{}(?=\])?(?!\d)'.format(re.escape(string1))

    for s in string2:
        if not isinstance(s, str):
            continue
```

```
        match = re.search(pattern, s)
        if match is not None:
            return True

    return False




import platform
try:
    import win32com.client
    win32com_available = True
except ModuleNotFoundError:
    win32com_available = False
import subprocess

def closeReports():
    try:
        if platform.system() == 'Windows' and win32com_available:
            word = win32com.client.Dispatch("Word.Application")
            for doc in word.Documents:
                doc.Close()
            word.Quit()
        elif platform.system() == 'Darwin':
            # Use subprocess to control Microsoft Word on macOS
            subprocess.call(['osascript', '-e', 'tell application "Microsoft Word" to quit'])
    except Exception as e:
        # Log an error message
        logging.error('closeDocuments(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('closeDocuments(): PASS')

def closeExcelWorkbooks():
    try:
        if platform.system() == 'Windows' and win32com_available:
            excel = win32com.client.Dispatch("Excel.Application")
            for book in excel.Workbooks:
                book.Close(SaveChanges=False)
            excel.Quit()
        elif platform.system() == 'Darwin':
            # Use subprocess to control Microsoft Excel on macOS
            subprocess.call(['osascript', '-e', 'tell application "Microsoft Excel" to quit'])
```

```
    except Exception as e:
        # Log an error message
        logging.error('closeExcelWorkbooks(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('closeExcelWorkbooks(): PASS')


def remove_empty_values(d):
    """
    This function takes a dictionary 'd' as input and removes all key-value pairs from it where the
        value is an empty string, whitespace, or an empty list.
    """
    keys_to_remove = []
    for key, value in d.items():
        if isinstance(value, list):
            if not value:
                keys_to_remove.append(key)
            else:
                for v in value:
                    if v.strip() == "":
                        keys_to_remove.append(key)
                        break
        elif value.strip() == "":
            keys_to_remove.append(key)
    for key in keys_to_remove:
        del d[key]



def createExcel3():

    global wb3
    wb3 = xw.Book()
    excelReport3 = wb3.sheets[0]
    excelReport3.name = "ReportNew"

    # Adding First Generation Tags header
    excelReport3.range("A1").value = 'First Generation'
    excelReport3.range("A1").font.Size = 14 # Change font size
    excelReport3.range("A1").font.ColorIndex = 2 # Change font color
    excelReport3.range('A1').font.bold = True
    excelReport3.range('A1:A1').color = (121,205,205) # Change cell background color
```

```
    # Adding Second Generation Tags header
    excelReport3.range("B1").value = 'Second Generation'
    excelReport3.range("B1").font.Size = 14 # Change font size
    excelReport3.range('B1').font.bold = True
    excelReport3.range("B1").font.ColorIndex = 2 # Change font color
    excelReport3.range('B1:B1').color = (121,205,205) # Change cell background color

    # Adding Third Generation Tags header
    excelReport3.range("C1").value = 'Third Generation'
    excelReport3.range("C1").font.Size = 14 # Change font size
    excelReport3.range('C1').font.bold = True
    excelReport3.range("C1").font.ColorIndex = 2 # Change font color
    excelReport3.range('C1:C1').color = (121,205,205) # Change cell background color

    # Adding Fourth Generation Tags header
    excelReport3.range("D1").value = 'Fourth Generation'
    excelReport3.range("D1").font.Size = 14 # Change font size
    excelReport3.range('D1').font.bold = True
    excelReport3.range("D1").font.ColorIndex = 2 # Change font color
    excelReport3.range('D1:D1').color = (121,205,205) # Change cell background color

    # Adding Fifth Generation Tags header
    excelReport3.range("E1").value = 'Fifth Generation'
    excelReport3.range("E1").font.Size = 14 # Change font size
    excelReport3.range('E1').font.bold = True
    excelReport3.range("E1").font.ColorIndex = 2 # Change font color
    excelReport3.range('E1:E1').color = (121,205,205) # Change cell background color

    # Adding Sixth Generation Tags header
    excelReport3.range("F1").value = 'Sixth Generation'
    excelReport3.range("F1").font.Size = 14 # Change font size
    excelReport3.range('F1').font.bold = True
    excelReport3.range("F1").font.ColorIndex = 2 # Change font color
    excelReport3.range('F1:F1').color = (121,205,205) # Change cell background color

    # Adding Fifth Generation Tags header
    excelReport3.range("G1").value = 'Seventh Generation'
    excelReport3.range("G1").font.Size = 14 # Change font size
    excelReport3.range('G1').font.bold = True
    excelReport3.range("G1").font.ColorIndex = 2 # Change font color
    excelReport3.range('G1:G1').color = (121,205,205) # Change cell background color

    # counters for Excel report3
```

```
    global counter1
    counter1 = 2
    global counter2
    counter2 = 1
    global counter3
    counter3 = 0
    global cell
    cell = 0;
    global cell2
    cell2 = 0;
    #print("dicts10",dicts10)
    remove_empty_values(dicts10)
    #print("new dicts10",dicts10)
    #print("this is dicts10",dicts10)

    hello = 'PUMP:RISK:10'


    for orpha in orphanChildren2Copy:
        if orpha != orphanChildren2Copy[0]:
            cell10 = str('A'+ str(counter1-1))
            excelReport3.range(cell10).value = 'SEPARATOR'
            excelReport3.range(cell10).font.Size = 14 # Change font size
            excelReport3.range(cell10).font.ColorIndex = 2 # Change font color
            cell11 = str(str(cell10) + ':G' + str(counter1-1))
            excelReport3.range(cell11).color = (178,223,238) # Change cell background color

        cell = str('A'+ str(counter1))
        cell2 = str(str(orpha))
        excelReport3.range(cell).value = cell2
        counter1 += 2 # counter for excel report
        counter2 += 1 # counter for excel report
        #print("hello",len(hello))
        #print(type(orpha))
        hx = str(orpha)
        #hx.replace(" ", "")
        #print(repr(hx))
        #print(hx)
        hx = hx.strip('\n')  # remove newline characters
        hx = hx.strip()     # remove leading/trailing whitespaces
        #print(repr(hx))
        #print(hx)
        #print(type(hx))
        hx.strip()
```

```
#print(len(hx))
keys = [h for h, v in dicts10.items() if check_string(hx, str(v))]
#wi = 'PUMP:RISK:10'
#keys3 = [h for h, v in dicts10.items() if check_string(wi, str(v))] # finds all the child tags
#print("keys3",keys3)

#print("keys",keys)
for item in keys: #keys are child tags of hx/the parent tag
    #print("child",item)
    if item != "" and item!= " ": # if the child tag is not empty

      counter2 = counter1 - 1
      cell = str('B'+ str(counter2))
      cell2 = str(item)
      #print("cell",cell)
      #print("cell2",cell2)
      excelReport3.range(cell).value = cell2
      counter2 += 1
      counter1 += 1
      stringKey = str(item)
      stringKey2 = (stringKey.replace(' ', ''))
      #duplicates2 = []
      #print("item is not " ", item)
      if str(stringKey2) in dicts10: # if the key is in the dictionary
      #for key in dicts10:
        if item in dicts10:
          #print("item is in dicts10",item)
          hx = item

          keys = [h for h, v in dicts10.items() if check_string(hx, v)] # finds all the child tags

          for item1 in keys: #keys are child tags of hx/the parent tag
            #print("grandchild",item1)
            if item1 != "" and item1!= " ":
              #if item1 not in duplicates2:
                #duplicates2.append(item1)
                counter2 = counter1 - 1
                cell = str('C'+ str(counter2))
                cell2 = str(item1)
                excelReport3.range(cell).value = cell2
                counter2 += 1
                counter1 += 1
                stringKey = str(item1)
                stringKey2 = (stringKey.replace(' ', ''))
```

```
                        #duplicates2 = []
                        #print("item is not " ", item)
                        if str(stringKey2) in dicts10: # if the key is in the dictionary
                        #for key in dicts10:
                            if item1 in dicts10:
                                #print("item is in dicts10",item)
                                hx = item1

                                keys = [h for h, v in dicts10.items() if check_string(hx, v)] # finds all
the child tags

                                for item2 in keys: #keys are child tags of hx/the parent tag
                                    #print("grandgrandchild",item2)
                                    if item2 != "" and item2!= " ":
                                        #if item2 not in duplicates2:
                                            #duplicates2.append(item1)
                                            counter2 = counter1 - 1
                                            cell = str('D'+ str(counter2))
                                            cell2 = str(item2)
                                            excelReport3.range(cell).value = cell2
                                            counter2 += 1
                                            counter1 += 1
                                            stringKey = str(item2)
                                            stringKey2 = (stringKey.replace(' ', ''))
                                            #duplicates2 = []
                                            #print("item is not " ", item)
                                            if str(stringKey2) in dicts10: # if the key is in the dictionary
                                            #for key in dicts10:
                                                if item2 in dicts10:
                                                    #print("item is in dicts10",item)
                                                    hx = item2

                                                    keys = [h for h, v in dicts10.items() if check_string(hx,
v)] # finds all the child tags

                                                    for item3 in keys: #keys are child tags of hx/the parent
tag
                                                        #print("grandgrandGrandchild",item3)
                                                        if item3 != "" and item3!= " ":
                                                            #if item2 not in duplicates2:
                                                                #duplicates2.append(item1)
                                                                counter2 = counter1 - 1
                                                                cell = str('E'+ str(counter2))
                                                                cell2 = str(item3)
```

```
                                        excelReport3.range(cell).value = cell2
                                        counter2 += 1
                                        counter1 += 1
                                        stringKey = str(item3)
                                        stringKey2 = (stringKey.replace(' ', ''))
                                        #duplicates2 = []
                                        #print("item is not " ", item)
                                        if str(stringKey2) in dicts10: # if the key is in the
dictionary
                                        #for key in dicts10:
                                            if item3 in dicts10:
                                                #print("item is in dicts10",item)
                                                hx = item3

                                                keys = [h for h, v in dicts10.items() if
check_string(hx, v)] # finds all the child tags

                                                for item4 in keys: #keys are child tags of
hx/the parent tag

#print("grandgrandGrandGrandchild",item3)
                                                    if item4 != "" and item4!= " ":
                                                        #if item2 not in duplicates2:
                                                            #duplicates2.append(item1)
                                                            counter2 = counter1 - 1
                                                            cell = str('F'+ str(counter2))
                                                            cell2 = str(item4)
                                                            excelReport3.range(cell).value =
cell2

                                                            counter2 += 1
                                                            counter1 += 1
                                                            stringKey = str(item4)
                                                            stringKey2 = (stringKey.replace(' ',
''))

                                                            #duplicates2 = []
                                                            #print("item is not " ", item)
                                                            if str(stringKey2) in dicts10: # if the
key is in the dictionary

                                                            #for key in dicts10:
                                                                if item4 in dicts10:
                                                                    #print("item is in dicts10",item)
                                                                    hx = item4
```

```
                                                    keys = [h for h, v in
dicts10.items() if check_string(hx, v)] # finds all the child tags

                                                    for item5 in keys: #keys are
child tags of hx/the parent tag

#print("grandgrandGrandGrandchild",item3)
                                                        if item5 != "" and item5!= " ":
                                                            #if item2 not in
duplicates2:

#duplicates2.append(item1)
                                                                counter2 = counter1 -
1
                                                                cell = str('G'+
str(counter2))
                                                                cell2 = str(item5)

excelReport3.range(cell).value = cell2
                                                                counter2 += 1
                                                                counter1 += 1
                                                                stringKey = str(item5)
                                                                stringKey2 =
(stringKey.replace(' ', ''))
                                                                #duplicates2 = []
                                                                #print("item is not " ",
item)
                                                                if str(stringKey2) in
dicts10: # if the key is in the dictionary
                                                                #for key in dicts10:
                                                                    if item5 in dicts10:
                                                                        #print("item is in
dicts10",item)
                                                                        hx = item5

                                                                        keys = [h for h, v
in dicts10.items() if check_string(hx, v)] # finds all the child tags

                                                                        for item6 in keys:
#keys are child tags of hx/the parent tag

#print("grandgrandGrandGrandchild",item3)
                                                                            if item6 != ""
and item6!= " ":
```

```
                                                                                #if item2 not
        in duplicates2:

        #duplicates2.append(item1)
                                                                                    counter2 =
        counter1 - 1
                                                                                    cell =
        str('H'+ str(counter2))
                                                                                    cell2 =
        str(item6)

        excelReport3.range(cell).value = cell2
                                                                                    counter2
        += 1
                                                                                    counter1
        += 1



      excelReport3.autofit()
   wb3.save('excelNew.xlsx') # Saving excel report as 'AllTags.xlsx'


def guiTree():
   family_trees = []
   current_tree = []

   for orpha in orphanChildren2Copy:
      if orpha != orphanChildren2Copy[0]:
         family_trees.append(current_tree)
         current_tree = []

      hx = orpha.strip()
      current_tree += " ├─ " + hx + "\n"

      keys = [h for h, v in dicts10.items() if check_string(hx, str(v))]

      for i, item in enumerate(keys): #keys are child tags of hx/the parent tag
         if item != "" and item != " ":
            current_tree += "│   ├─ " + item + "\n"
            stringKey = item.strip()
            if stringKey in dicts10:
               hx = item
               keys1 = [h for h, v in dicts10.items() if check_string(hx, v)]
```

```
            for j, item1 in enumerate(keys1):
                if item1 != "" and item1 != " ":
                    current_tree += "│   │   ├── " + item1 + "\n"
                    stringKey = item1.strip()
                    if stringKey in dicts10:
                        hx = item1
                        keys2 = [h for h, v in dicts10.items() if check_string(hx, v)]
                        for k, item2 in enumerate(keys2):
                            if item2 != "" and item2 != " ":
                                current_tree += "│   │   │   ├── " + item2 + "\n"
                                stringKey = item2.strip()
                                if stringKey in dicts10:
                                    hx = item2
                                    keys3 = [h for h, v in dicts10.items() if check_string(hx, v)]
                                    for l, item3 in enumerate(keys3):
                                        if item3 != "" and item3 != " ":
                                            current_tree += "│   │   │   │   ├── " + item3 + "\n"
                                            stringKey = item3.strip()
                                            if stringKey in dicts10:
                                                hx = item3
                                                keys4 = [h for h, v in dicts10.items() if check_string(hx, v)]
                                                for m, item4 in enumerate(keys4):
                                                    if item4 != "" and item4 != " ":
                                                        current_tree += "│   │   │   │   │   ├── " + item4 + "\n"
                                                        stringKey = item4.strip()
                                                        if stringKey in dicts10:
                                                            hx = item4
                                                            keys5 = [h for h, v in dicts10.items() if
check_string(hx, v)]
                                                            for n, item5 in enumerate(keys5):
                                                                if item5 != "" and item5 != " ":
                                                                    current_tree += "│   │   │   │   │   │   ├── " + item5 +
"\n"
                                                                    stringKey = item5.strip()
                                                                    if stringKey in dicts10:
                                                                        hx = item5
                                                                        keys6 = [h for h, v in dicts10.items() if
check_string(hx, v)]
                                                                        for o, item6 in enumerate(keys6):
                                                                            if item6 != "" and item6 != " ":
                                                                                current_tree += "│   │   │   │   │   │   ├── "
+ item6 + "\n"
```

```
    family_trees.append(current_tree)
    return family_trees
```

Gui.py

```
import logging
import docx
from docx import Document
from docx.shared import RGBColor
from docx.shared import Inches
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from tkinter import ttk
from typing import Tuple


from tkinter import scrolledtext
from tkinter.scrolledtext import ScrolledText
import re
import copy
import time


# This libraries are for opening word document automatically
import os
import platform
import subprocess


# This library is for opening excel document automatically
import xlwings as xw
import pandas as pd
import matplotlib.pyplot as plt
```

```
import Targest2


import Targest
global scrolled_text_box


from tkinter import messagebox


import webbrowser



def GUI1():

    try:
        # Creates the gui
        window = Tk(className=' TARGEST v.1.18.1 ')
        # set window size #
        window.geometry("1000x620")
        window['background']='#009ce8'


        #canvas = Canvas(window, width=1000, height=620)
        #canvas.pack()


        # Create a horizontal gradient
        #for i in range(1000):
        #    r = int(i/1000 * 152)
        #    g = 75  # fixed green value
        #    b = 255 - int(i/1000 * 108)
        #    color = (0, 156, 232)
            #color = '#{:02x}{:02x}{:02x}'.format(r, g, b)
```

```
    #    canvas.create_rectangle(i, 0, i+1, 1000, fill=color, outline='')


    icon = PhotoImage(file='TARGEST.png')
    window.iconphoto(True, icon)


    # Create a style for the widgets
    style = ttk.Style()
    #style.configure('Emergency.TButton', font='helvetica 24', foreground='red', padding=10)
    style.configure("TButton", font=("Segoe UI", 10, "bold"), background="#b2d8ff",
foreground="black")



    # Create a canvas
    #canvas = tk.Canvas(window, width=200, height=400)
    #canvas.place(x=500, y=500)
    #canvas.pack(side='left')



    # Load your anime figures as image files
    #figure1 = tk.PhotoImage(file='TARGEST2.png', width=200, height=400)
    #figure2 = tk.PhotoImage(file='eren.png')


    # Place your anime figures on the canvas
    #canvas.create_image(100, 100, image=figure1)


    # button 1
    ttk.Button(window, text="Choose list of Documents", command=Targest2.generateReport,
width = 22).place(x=136, y=10)


    # button 2
    global genRep
```

```
    genRep = ttk.Button(window, text="Generate Reports", state= DISABLED,
command=Targest2.generateReport2, width = 22)
    genRep.place(x=330, y=10)


    # button 3
    global allTagsButton
    allTagsButton = ttk.Button(text="Open All Tags Table Report", state= DISABLED,
command=Targest2.getDocumentTable, width = 34)
    allTagsButton.place(x=627, y=10)


    # button 4
    global getDoc
    getDoc = ttk.Button(window, text="Open Child and Parent Tags Report", state=
DISABLED, command=Targest2.getDocument, width = 34)
    getDoc.place(x=627, y=35)


    # button 5
    global getOrphanDoc
    getOrphanDoc = ttk.Button(text="Open Orphan Tags Report", state= DISABLED,
command=Targest2.getOrphanDocument, width = 34)
    getOrphanDoc.place(x=627, y=60)


    # button 6
    global getChildlessDoc
    getChildlessDoc = ttk.Button(text="Open Childless Tags Report", state= DISABLED,
command=Targest2.getChildlessDocument, width = 34)
    getChildlessDoc.place(x=627, y=85)


    # button 7
    global getTBVdoc
    getTBVdoc = ttk.Button(text="Open TBV Word Report", state= DISABLED,
command=Targest2.getTBV, width = 34)
    getTBVdoc.place(x=627, y=110)
```

```
    # button 8
    global getTBDdoc
    getTBDdoc = ttk.Button(text="Open TBD Word Report", state= DISABLED,
command=Targest2.getTBD, width = 34)
    getTBDdoc.place(x=627, y=135)


    # button 9
    global getExcel
    getExcel = ttk.Button(text="Open Requirements Excel Report", state= DISABLED,
command=Targest2.createExcel, width = 34)
    getExcel.place(x=627, y=160)


    # button 10
    #global getExcel2
    #getExcel2 = ttk.Button(text="Open All Tags Excel Report", state= DISABLED,
command=Targest2.createExcel2, width = 30)
    #getExcel2.place(x=620, y=185)


    # button 10
    global getExcel2
    getExcel2 = ttk.Button(text="Open Relationship Trees Excel Report", state= DISABLED,
command=Targest2.createExcel3, width = 34)
    getExcel2.place(x=627, y=185)

    # button 10
    global TreeDiagram
    TreeDiagram = ttk.Button(text="Create Family Trees", state= DISABLED, command
=lambda: Targest.text3(window), width = 34)
    TreeDiagram.place(x=627, y=210)


    # button 11
```

```
    global Website
    Website = ttk.Button(text="Visit our Website", state= ACTIVE, command =lambda:
open_website(), width = 30)
    Website.place(x=205, y=62)



    #global button
    #button = Button(text="End Program", command=window.destroy, width = 30,
font=("Segoe UI", 10), background="#4CAF50", foreground="white")
    #button.place()



    # button 11
    global button
    button = ttk.Button(text="End Program", command=lambda:[window.destroy(),
Targest2.closeReports(), Targest2.closeExcelWorkbooks()], width = 30)
    button.place(x=205, y=38)



    # Create text widget and specify size.
    global Txt
    Txt = ScrolledText(window, wrap=tk.WORD, height = 30, width = 60)
    Txt.place(x=25, y=120)
    Txt.configure(bg='grey', fg='white')



    # Create a label for the developers
    labelDevs = Label(window, text="Developers:\nJan William Haug\nAdrian
Bernardino\nStephania Rey", font=("Segoe UI", 10, "bold"), bg="#E5CCFF")
    labelDevs.place(x=690, y=510)
    labelDevs.config(borderwidth=2, relief="groove", padx=10, pady=5, fg="black")



    # Create ScrolledText widget
    scrolled_text_box = ScrolledText(window, wrap=tk.WORD, height=15, width=47)
```

```
        scrolled_text_box.place(x=566, y=240)
        scrolled_text_box.configure(bg='grey', fg='white')


        # Load the image file
        global imageLogo
        imageLogo = PhotoImage(file="TARGEST3.png")


        # Create a label to display the image
        label2 = Label(window, image=imageLogo)
        label2.place(x=25, y=10)



        msg3 = ('You need a text file with paths to your documents\n 1. Please choose your
documents by clicking on \n    the "Choose list of Documents" button.\n 2. Once the documents
are displayed, Click "Generate Reports"\n\n')
        Txt.insert(tk.END, msg3) #print in GUI


        # show a pop-up message
        #messagebox.showinfo("Welcome to TARGEST",  "Make sure you have closed all your
previous Word Reports and Excel Reports, before running this application")
        messagebox.showinfo("Welcome to TARGEST",  "Make sure to save a text file with the
paths to the documents you want to use, if you haven't already")


    except Exception as e:
        # Log an error message
        logging.exception('main(): ERROR', exc_info=True)
    else:
        # Log a success message
        logging.info('main(): PASS')


        window.mainloop()
```

```
def open_website():
    webbrowser.open("https://targest-website.vercel.app/")
```

TE System Testing(Test Case Design)

We performed 5 different test cases while working on the program. To start with the testing phase of our system, we decided to write our test cases based on scenarios we would have. The test cases gives us the assurance that the flow of the system is working fine. When our test cases failed, they helped us notice any bugs our system had, and that the system needed to be fixed. If the test cases passed, that meant that the flow is working as expected and the system were working and behaving as per the requirement. The test cases was performed using the IDE PyCharm, Command Prompt on windows and terminal on Mac.

Table 8.2.1. Test Suite TS-001: Child Tags

Test Case ID	Test Stage	Test Case Description	Tested
TC-001	Unit	If The Child-tags are being printed out	Yes
TC-009	Unit	To see the leading tags list and their expected output	No

Table 8.2.2. Test Suite TS-002: Parent Tags

Test Case ID	Test Stage	Test Case Description	Tested
TC-002	Unit	Check if all parent tags are being printed out	Yes
TC-006	Unit	involves the creation of a class that performs two testing functions after generating a report.	No
TC-007	Unit	focused on the list of lists provided in the first document	No

Table 8.2.3. Test Suite TS-003: TC-003

Test Case ID	Test Stage	Test Case Description	Tested
TC-003	Unit	Open generated report automatically	Yes
TC-008	Unit	test to display orphan tags list	No

Table 8.2.4. Test Suite TS-004: Child Tags, with at least one parent

Test Case ID	Test Stage	Test Case Description	Tested
TC-004	Unit	Display only child tags with a parent, and not orphan tags	Yes
TC-010	Unit	to show the childless tags	No

Table 8.2.5. Test Suite TS-005: TC-005

Test Case ID	Test Stage	Test Case Description	Tested
TC-005	Acceptance	As a new user I'm able to open the program and run it.	Yes

The first test case, were testing that all the child tags in the documents provided would be displayed when running the program in terminal at this date.

Table 8.2.6. Test Case TC-001

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-001: Child Tags	
Test Case ID	TC-001 (Unit Test)	
What To Test	If The Child-tags are being printed out	
Test Data Input	Word Document/Documents provided by the user	
Expected Result	All the child tags will be found in the document and printed out	
Traceability	Relevant User Req.(s)	UF-B,UF-C,UF-E,UF-H,UP-01
	Relevant System Req.(s)	SF-A-01,SF-C-01,SF-D-01,SF-E-01,SF-F-01,SF-G-01,SF-H-01,SO-01-01,SP-01-01
	Relevant Use Case(s)	UC-001,UC-003,UC-004,UC-006,UC-007

Acknowledgment: Generated from the CapStone process management system ©2023

Table 8.2.6. Test Case TC-001

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-001: Child Tags	
Test Case ID	TC-001 (Unit Test)	
What To Test	If The Child-tags are being printed out	
Test Data Input	Word Document/Documents provided by the user	
Expected Result	All the child tags will be found in the document and printed out	
Traceability	Relevant User Req.(s)	UF-B,UF-C,UF-E,UF-H,UP-01
	Relevant System Req.(s)	SF-A-01,SF-C-01,SF-D-01,SF-E-01,SF-F-01,SF-G-01,SF-H-01,SO-01-01,SP-01-01
	Relevant Use Case(s)	UC-001,UC-003,UC-004,UC-006,UC-007
Acknowledgment: Generated from the CapStone process management system ©2023		

The second test case checked the same as the first test case, except that it made sure that all the correct parent tags form the documents were displayed.

Table 8.2.8. Test Case TC-003

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-003: TC-003	
Test Case ID	TC-003 (Unit Test)	
What To Test	Open generated report automatically	
Test Data Input	getDocument function was created in order to use the open document button	
Expected Result	The generated report is able to open automatically	
Traceability	Relevant User Req.(s)	
	Relevant System Req.(s)	SF-B-01,SP-01-01
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2023		

The third test case, was for our newly created “getDocument” function. This function was made to open the report created faster and more efficient. We had to make sure that that it was opening the right word document, and had no errors or bugs.

Table 8.2.9. Test Case TC-004

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-004: Child Tags, with at least one parent	
Test Case ID	TC-004 (Unit Test)	
What To Test	Display only child tags with a parent, and not orphan tags	
Test Data Input	Word Documents	
Expected Result	Display only child tags with a parent, and not orphan tags	
Traceability	Relevant User Req.(s)	UF-B,UF-H,UP-01
	Relevant System Req.(s)	SF-B-01,SF-H-01,SO-01-01,SP-01-01
	Relevant Use Case(s)	UC-001,UC-003,UC-004
Acknowledgment: Generated from the CapStone process management system ©2023		

The fourth test case was created to make sure that our program was able to spot the child tags that had a parent, and to exclude the orphan tags.

Table 8.2.15. Test Case TC-005

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-005: TC-005	
Test Case ID	TC-005 (Acceptance Test)	
What To Test	As a new user I'm able to open the program and run it.	
Test Data Input	Tom would choose a document from his own directory.	
Expected Result	Have a report be generated as expected with parent tags and child tags organized.	
Traceability	Relevant User Req.(s)	UF-A,UF-B,UF-D,UF-F,UF-G,UP-01
	Relevant System Req.(s)	SF-A-01,SF-B-01,SF-B-02,SF-B-03,SF-F-01,SO-01-01,SP-01-01
	Relevant Use Case(s)	UC-001,UC-004,UC-005
Acknowledgment: Generated from the CapStone process management system ©2023		

The fifth test case was performed by our industry mentor Tom, and this test was to make sure that our program would be able to run on his computer with his documents.

The 6th test case was created to make sure that our program was able to spot all parent tags and print them out.

Table 8.2.10. Test Case TC-006

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-002: Parent Tags	
Test Case ID	TC-006 (Unit Test)	
What To Test	involves the creation of a class that performs two testing functions after generating a report.	
Test Data Input	first function retrieves a collection of file names and compares them with the expected output list provided in the image	
Expected Result	expected output lit	
Traceability	Relevant User Req.(s)	UF-C,UO-01
	Relevant System Req.(s)	
	Relevant Use Case(s)	UC-004
Acknowledgment: Generated from the CapStone process management system ©2023		

The 7th test case was to test if all parent tags are identified along with their corresponding child tags.

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-002: Parent Tags	
Test Case ID	TC-007 (Unit Test)	
What To Test	focused on the list of lists provided in the first document	
Test Data Input	examines all the parent tags that are correctly identified and reads their corresponding child elements	
Expected Result	compares the resulting list of parent elements to the expected output.	
Traceability	Relevant User Req.(s)	
	Relevant System Req.(s)	SF-B-03,SF-H-01
	Relevant Use Case(s)	UC-003,UC-005,UC-007
Acknowledgment: Generated from the CapStone process management system ©2023		

The 8th test case was to test if all orphan tags are identified and shown as a list.

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-003: TC-003	
Test Case ID	TC-008 (Unit Test)	
What To Test	test to display orphan tags list	
Test Data Input	test documents to look for orphan tags	
Expected Result	expected output orphan tags list	
Traceability	Relevant User Req.(s)	UF-E,UO-01
	Relevant System Req.(s)	SF-B-03,SF-E-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2023		

Table 8.2.13. Test Case TC-009

The 9th test case was to test if all child tags are identified and shown as a list.

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-001: Child Tags	
Test Case ID	TC-009 (Unit Test)	
What To Test	To see the leading tags list and their expected output	
Test Data Input	parse and search through documents to display child tags	
Expected Result	show expect child tags list	
Traceability	Relevant User Req.(s)	UO-01,UP-01
	Relevant System Req.(s)	SF-B-03,SP-01-01
	Relevant Use Case(s)	UC-003,UC-005
Acknowledgment: Generated from the CapStone process management system ©2023		

Table 8.2.14. Test Case TC-010

The 10th test case was to test if all childless tags are identified and shown as a list.

Project Name:	Requirements Tracing/Management Software	
Test Suite	TS-004: Child Tags, with at least one parent	
Test Case ID	TC-010 (Unit Test)	
What To Test	to show the childless tags	
Test Data Input	enter function, look through documents and variable to see childless tags	
Expected Result	show expected out of childless tags	
Traceability	Relevant User Req.(s)	UF-D
	Relevant System Req.(s)	SF-B-03,SF-D-01
	Relevant Use Case(s)	UC-003
Acknowledgment: Generated from the CapStone process management system ©2023		

Our first execution report, first identified a defect on 10/29/2022 by Jan, but was later fixed on 10/29/2022, and passed the test.

Table 8.3.1. Execution Report of Test Case TC-001

Project Name:	**Requirements Tracing/Management Software**					
Test Case ID:	TC-001					
Testing Tools Used:	PyCharm					
Testing Type:	Functional testing					
Execution Steps:	1	Run program				
	2	Choose Word Document				
	3	Push the Generate Report Document button				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Jan	10/28/2022	All the child tags were not correctly printed out form the chosen word document	Fail	problem with the find childTag function	10/29/2022 by Jan
2	Jan	10/29/2022	All the child tags were not correctly printed out form the chosen word document	Pass	No defect	10/29/2022 by Jan
Execution Summary:	Checked if the Child-tags were correctly being printed out, and fixed the bug we had.					
Acknowledgment: Generated from the CapStone process management system ©2022						

The second test case failed because regular expression method for two of the functions were missing. This defect was fixed by Stephania on 11/20/2022 and passed the test.

Table 8.3.2. Execution Report of Test Case TC-002

Project Name:	**Requirements Tracing/Management Software**					
Test Case ID:	TC-002					
Testing Tools Used:	Pycharm					
Testing Type:	Functional testing					
Execution Steps:	1	Run program				
	2	Check the terminal for word tags displayed				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	Stephania	10/22/2022	all parent tags were printed out	Pass	No defect	10/22/2022 by Stephania
Execution Summary:	Checked if all parent tags were printed out, and everything went was expected.					
Acknowledgment: Generated from the CapStone process management system ©2022						

The third test first failed because of a defect on 11/07/2022, the OS library was not compatible with MacOS. This defect was later fixed by using platform and subprocess on 11/08/2022 by Adrian.

Table 8.3.3. Execution Report of Test Case TC-003

Project Name:	Requirements Tracing/Management Software					
Test Case ID:	TC-003					
Testing Tools Used:	Pycharm, os, subprocess, platform					
Testing Type:	Functional testing					
Execution Steps:	1	Start program and choose from Menu				
	2	Choose Word Document				
	3	Push Generate Report Button				
	4	Push the Open Document Button				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Adrian	11/06/2022	Document wasn't opening automatically.	Fail	os library wasn't compatible with MacOS	11/07/2022 by Adrian
2	Adrian	11/08/2022	Document was able to open automatically	Pass	used platform and subprocess to have it work on MacOS	11/08/2022 by Adrian
Execution Summary:	Wanted this button to be compatible on both Windows and MacOS. Needed to create conditional statements to allow the document to open on either MacOS, Windows, or any other linux system.					
Acknowledgment: Generated from the CapStone process management system ©2022						

The fourth test case failed because regular expression method for two of the functions were missing. This defect was fixed by Stephania on 11/20/2022 and passed the test.

Table 8.3.4. Execution Report of Test Case TC-004

Project Name:	Requirements Tracing/Management Software					
Test Case ID:	TC-004					
Testing Tools Used:	Pycharm					
Testing Type:	Functional testing					
Execution Steps:	1	Run program				
	2	Generate a word document				
	3	Open the word document created				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Stephania	10/29/2022	all Child tags were printed out	Fail	Regular expressions method to both def GetParentTags(): and def GetChildTags(): functions were missi	11/30/2022 by Stephania
2	Stephania	10/29/2022	Only Child tags with a parent were printed	Pass	No defect	11/30/2022 by Stephania
Execution Summary:	Tried to display only child tags with a parent, and not orphan tags. Fixed the defect by adding regular expressions method to both def GetParentTags() and def GetChildTags() functions					
Acknowledgment: Generated from the CapStone process management system ©2022						

The fifth test case passed and was tested on 11/10/2022 by Tom.

Table 8.3.5. Execution Report of Test Case TC-005

Project Name:	Requirements Tracing/Management Software					
Test Case ID:	TC-005					
Testing Tools Used:	Word Docs, Terminal					
Testing Type:	Functional testing					
Execution Steps:	1	Open the downloaded zip file of program				
	2	Run the program on Terminal				
	3	Choose Word Document				
	4	Push Generate Report button				
	5	Push the Open Document button				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Tom	11/10/2022	Found duplicate tags that were not caught by user beforehand	Pass	Word Document had extra duplicate tags	11/10/2022 by Tom
Execution Summary:	I'm able to open the program and run my own documents and generate reports.					
Acknowledgment: Generated from the CapStone process management system ©2022						

Regarding the testing portion of our application, the team decided to create a series of types of tests. The purpose of these test types is to allow the team to see how the software is working by it detecting errors or bugs and showing the expected outputs as pass or fail. This strategy will be helpful when we jump onto using Pytest for unit testing. It will give us input on how we should go about adding additional code to the current software that will catch these errors if the same were to be entered manually. It also will display how the test cases are working in relation to the documents.

Additional Tests

Test 1
Purpose: simple test
Input: Using 5 documents with new tags created
Expected output: Leading tag and trailing tag correct and should pass

Test 2
Purpose: Test for space in between tags
Input: Have 3 tags with spaces between colon and tag somewhere
Expected output: Should ignore these spaces and should pass

Test 3
Purpose: Test if code can identify error in the tags
Input: Misspell 3 tags in the documents
Expected output: The code should list these tags as orphan tags

Test 4
Purpose: Identify tags that may look similar
Input: Tags that have a common name in it (ex. Similar to ACE:SRU:100 and BOLUS: SRS:100)
Expected output: The code should list these as different tags.

Test 5
Purpose: Identify leading tags that are missing trailing tags
Input: Tags may have leading tags, but no trailing tags similar to RISK.
Expected output: The code should either ignore these and return then in a separate part of the document

Test 6
Purpose: Test whether code can identify bracket tag errors

Input: Eliminate the ending or beginning bracket on some of the tags
Expected output: The code should ignore this and pass correctly

Test 7
Purpose: Stress test the code
Input: Combine many of the tests above the test the code
Expected output: All outcome above should not change

12.1 Glossary of Relevant Domain Terminology

TARGEST- which simply just stands for the name of the app "technical abstraction report generator extraction software tool".

Embed tags – Defines a container for an external resource, such as a web page, a picture, a media player, or a plug-in application.

Xlwings – An open source package that allows you to automate Excel with Python on Windows and macOS. Writes Excel macros and UDF's in Python.

PyUnit – A way to create unit testing programs and Unit Tests with Python.

Test cases – A document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

PyDoc – A module that automatically generates documentation from Python modules.

ProUnit – A framework to create automated unit tests for Progress 4GL platform.

4GL – A fourth-generation programming language.

Pandas is a software library written for python. Pandas is used for data manipulation and analysis.

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations.

13.0 Reference Documents

Arbuckle, D. (2010) *Python testing Beginner's Guide*. Birmingham: Packt Pub.

Docs.xlwings.org, “Python API, Top Level Functions”, version 0.27.15 https://docs.xlwings.org/en/0.27.15/api.html

Elder, John, “Creating Buttons With TKinter - Python Tkinter GUI Tutorial #3”, Codemy.com, January 14. 2019, https://www.youtube.com/watch?v=yuuDJ3-EdNQ.

Folkstalk.com, “Python Read Word Document With Code Examples”, July 2022, https://www.folkstalk.com/tech/python-read-word-document-with-code-examples/.

Jenn, Jia, “Automate Excel with Python”, September 2021, https://www.youtube.com/playlist?list=PL3JVwFmb_BnTPmU9Vax-Q1u2CQ9QbFmiJ .

Malik, Usman, “Reading and Writing MS Word Files in Python via Python-Docx Module”, March 2020,https://stackabuse.com/reading-and-writing-ms-word-files-in-python-via-python-docx-module/.

Pajankar, A. (2022). *Python unit test automation practical techniques for python developers and Testers*. Apress..

ProgrammingLanguage (2019) *Learn Pytest in 60 Minutes : Python Unit Testing Framework*. https://www.youtube.com/watch?v=bbp_849-RZ4&list=RDQMiQoK5eUGjwU&index=11

SourceMaking.com,“Strategy Design Pattern”, https://sourcemaking.com/design_patterns/strategy.

SourceMaking.com, “Facade Design Pattern”, https://sourcemaking.com/design_patterns/facade/python/1.

Sharma, Akshay, “How to create a GUI using the TKinter-library", September 2021, https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/.

YouTube. (2017). *Python Tutorial: Unit Testing Your Code with the unittest Module*. Python Tutorial: Unit Testing Your Code with the unittest Module

YouTube.(2022).*Pytest Unit Testing tutorial • how to test your python code*. YouTube. https://www.youtube.com/watch?v=YbpKMIUjvK8