

Ain Shams University  
Faculty of Engineering  
Computer and systems Department



## **CSE323 – Data Structures and Algorithms.**

### **Project- XML Editor**

**Submitted by:**

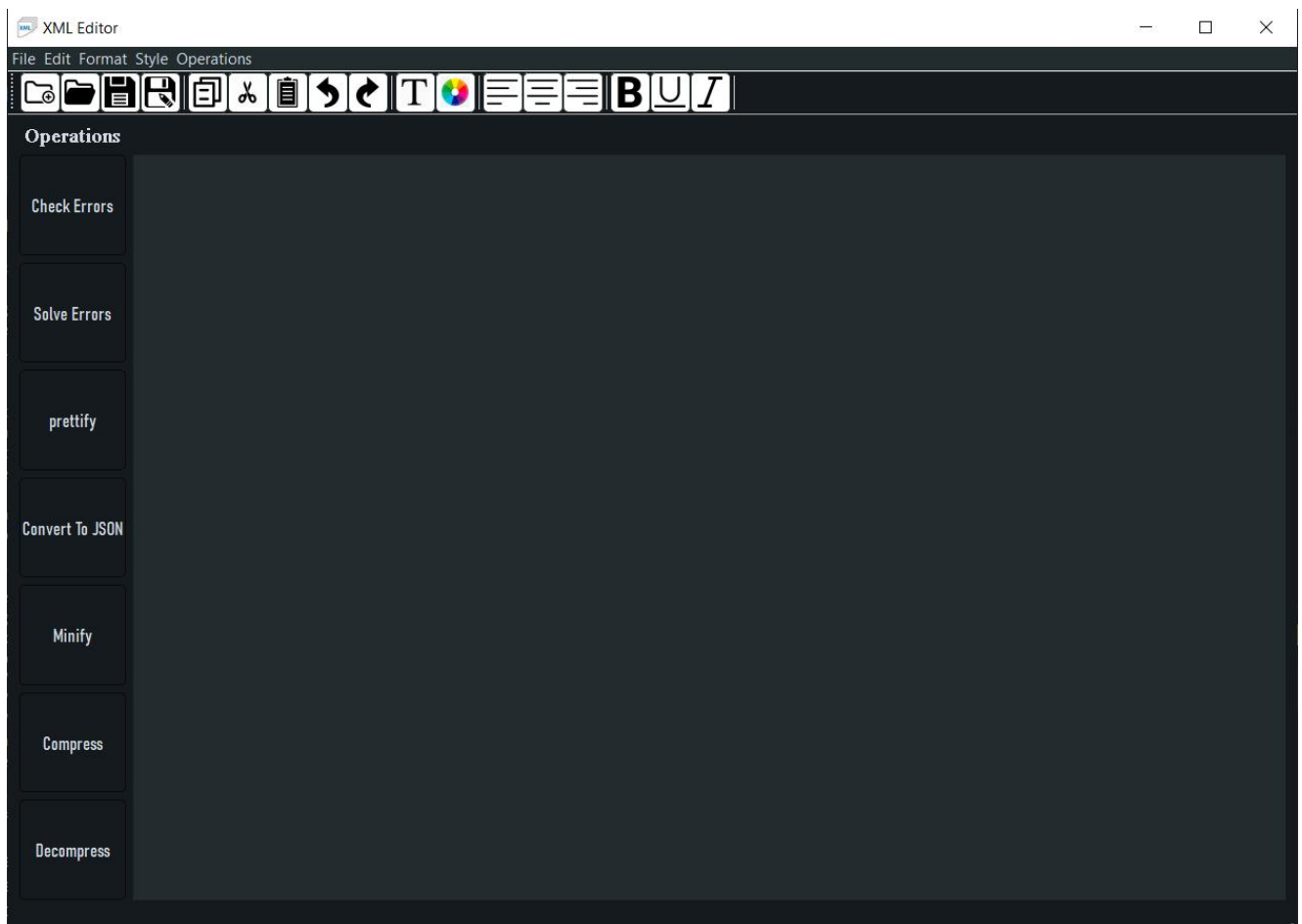
1600758	عبدالرحمن مصطفى فؤاد ابراهيم
1601626	نيفين نبيل شكري
1600007	ابراهيم جمال ابراهيم
1601227	محمد عبدالرحيم محمد صابر

# 1 CONTENTS

---

2	BACKGROUND AND GUI: .....	3
2.1	Basic Functions: .....	3
3	ALGORITHMS AND COMPLEXITY: .....	5
3.1	Check Errors: .....	5
3.1.1	Algorithm Explanation: .....	5
3.2	Prettify: .....	7
3.2.1	Complexity : $O(n^2)$ .....	7
3.2.2	Algorithm Explanation: .....	7
3.3	Convert to JSON: .....	8
3.3.1	Complexity : $O(n^2)$ .....	8
3.3.2	Algorithm Explanation: .....	8
3.4	Minify: .....	10
3.4.1	Complexity: $O(n)$ .....	10
3.5	Compress: .....	11
3.5.1	Complexity: $O(n \log n)$ .....	11
3.5.2	Compression Algorithm: .....	11
3.6	Decompression: .....	14
3.6.1	Complexity: $O(n)$ .....	14
4	GUI Error Handling: .....	16
5	References and resources: .....	17
6	LINK TO THE GITHUB REPOSITORY: .....	17
7	LINK TO THE YOUTUBE VIDEO: .....	17

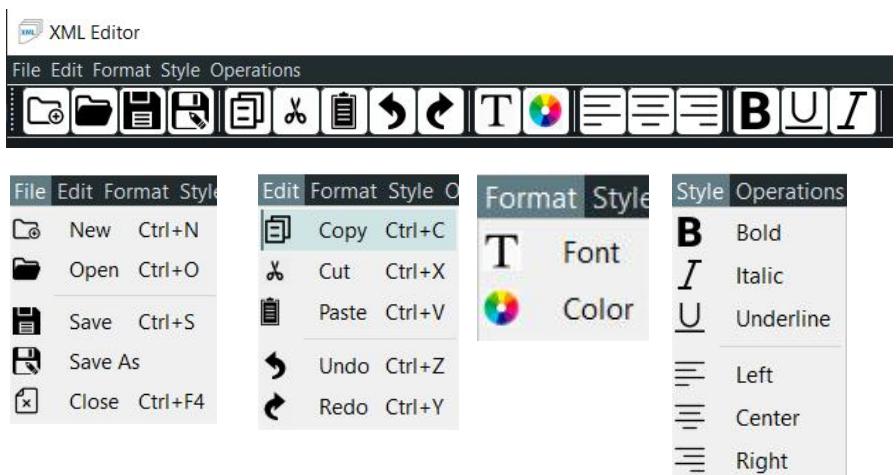
## 2 BACKGROUND AND GUI:



XML editor program is written in python using PyQt5 library. It has all the basic functions that a text editor would have, and can be used to edit (.txt , .xml , and .json) files.

### 2.1 BASIC FUNCTIONS:

You can access these basic functions either through the menu bar, or the shortcuts beneath it (tool bar.)



- **New File:**

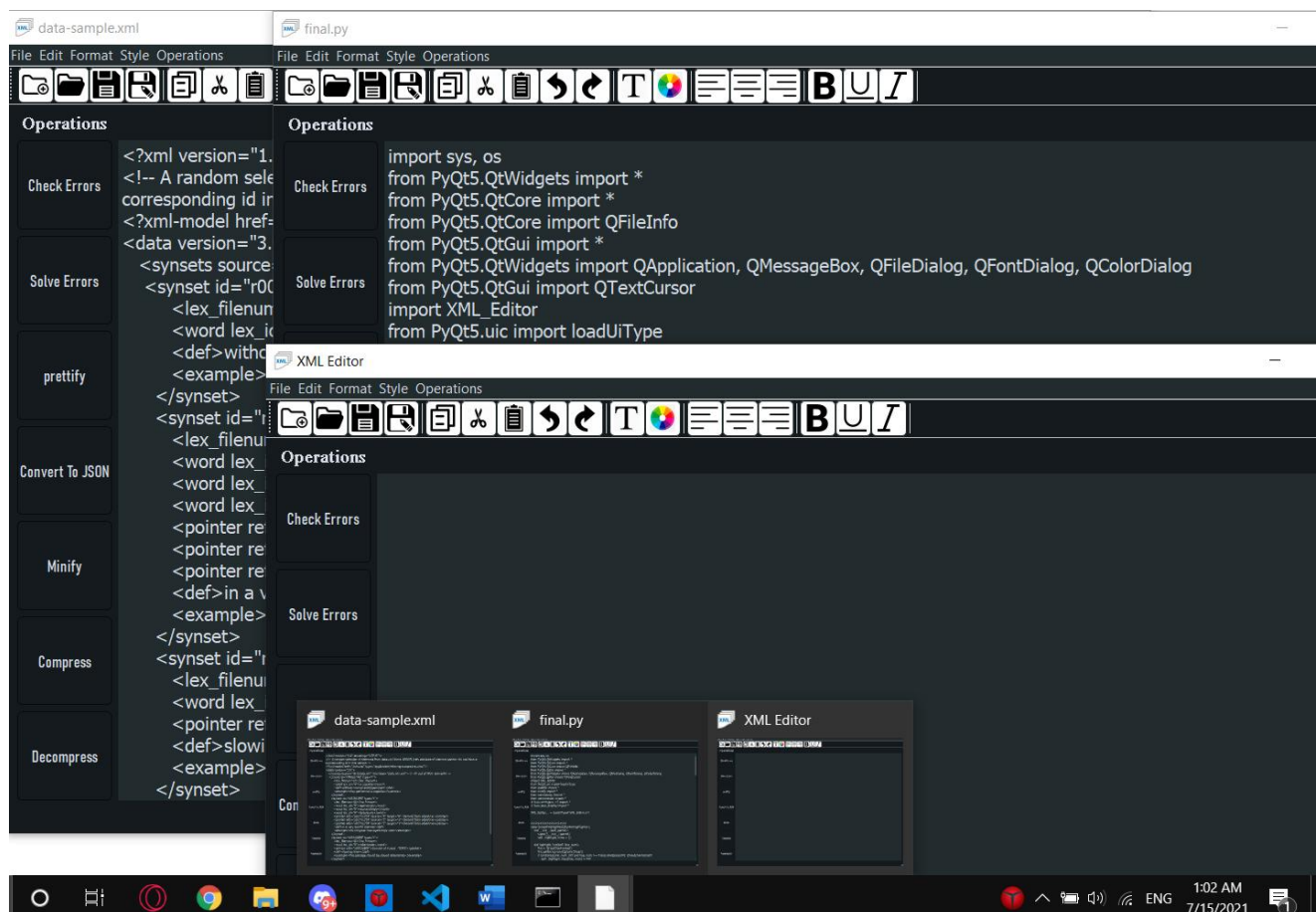
The user can use the program to create new files. The program provides the ability to create and operate on multiple files at the same time. Also, If the user did not save their actions on a new file, the program will prompt them to save before they exit.

- **Open File:**

The GUI allows the user to choose the file they desire. Also, it allows multiple files to be opened at the same time and the user can perform operations on them at the same time. The program changes the title of each window to the name of the current file.

- **Save File:**

The user can save the result of the operations at any time, either in the same file using (save) or in a different file using (save as) option.



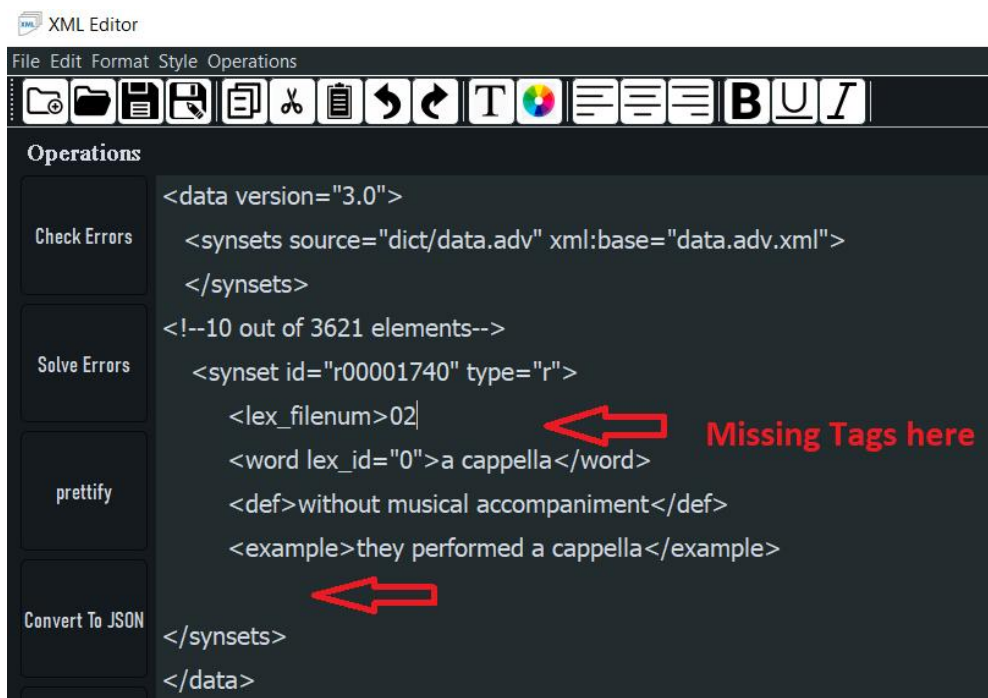
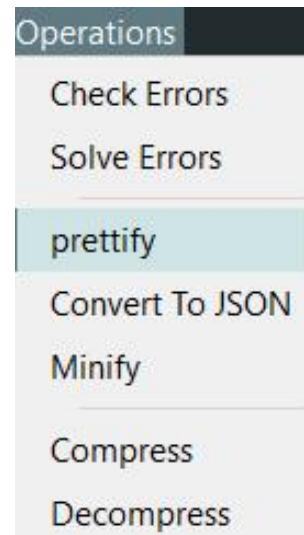
### 3 ALGORITHMS AND COMPLEXITY:

In addition to the basic functions, the program provides other operations on the files. The user can access these operations through “Operations” menu in the Menu bar, or through the quick access set of buttons on the left side.

#### 3.1 CHECK ERRORS:

**Complexity :  $O(n^2)$**

By pressing the “Check Errors” button, the GUI can detect the missing tags and solves them on its own.



#### 3.1.1 Algorithm Explanation:

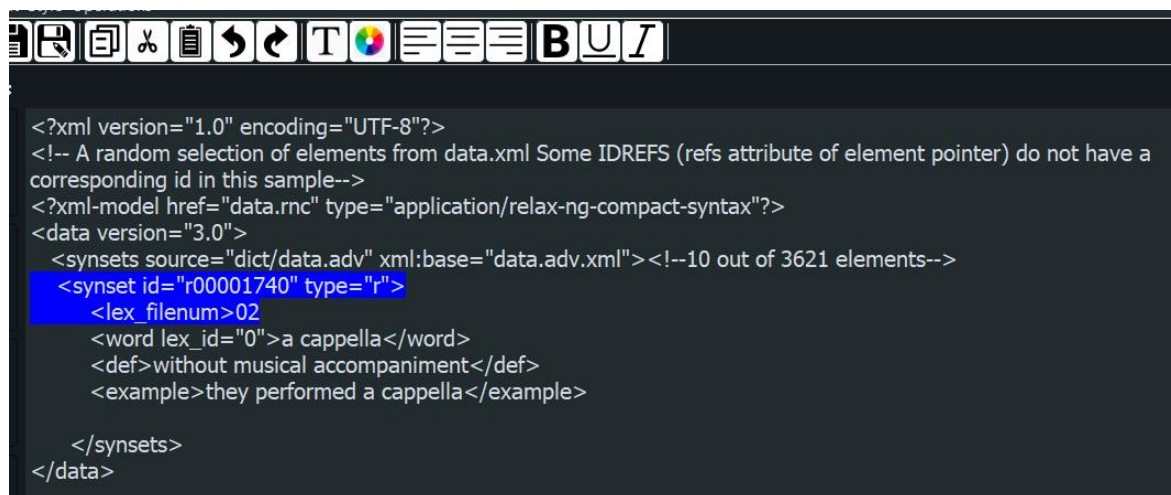
We have two buttons one for showing the error and one for solving them.

The idea is we loop over the given XML and store opening tags in a stack if we find a closing tag, we add it to another stack and compare the top of both stacks we pop the two stacks.

If the top in both stock is not equal, we highlight the errors for the user in the GUI, and then we fix them.

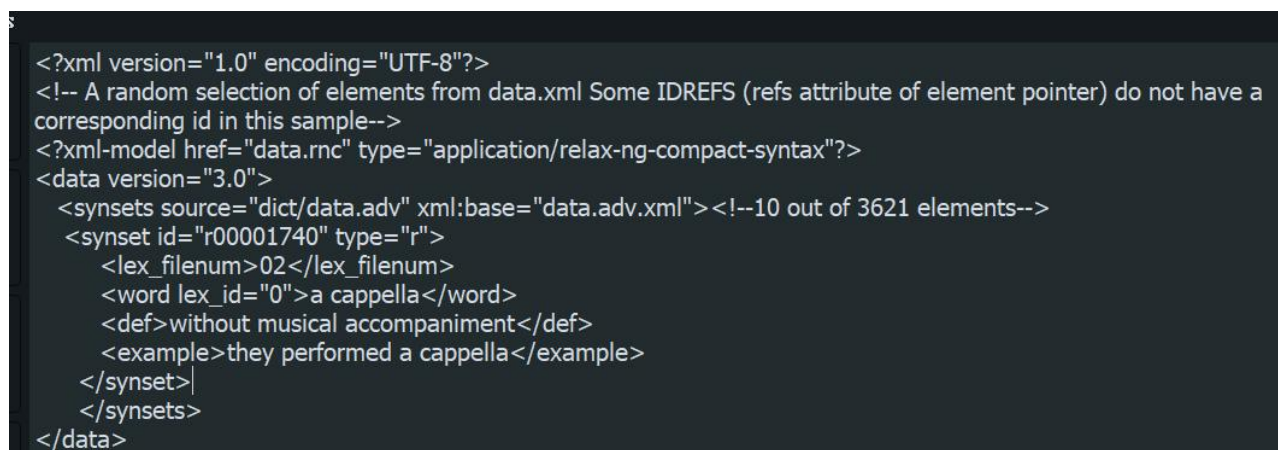
Note that, “Check Errors” highlights the opening tags that are missing closing tags and their parents, while “Solve Errors” will fix them.

The output of the “Check Errors” Button:



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A random selection of elements from data.xml Some IDREFS (refs attribute of element pointer) do not have a
corresponding id in this sample-->
<?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?>
<data version="3.0">
  <synsets source="dict/data.adv" xml:base="data.adv.xml"><!--10 out of 3621 elements-->
    <synset id="r00001740" type="r">
      <lex_filenum>02
      <word lex_id="0">a cappella</word>
      <def>without musical accompaniment</def>
      <example>they performed a cappella</example>
    </synsets>
  </data>
```

The output of the “Solve Errors” Button:



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A random selection of elements from data.xml Some IDREFS (refs attribute of element pointer) do not have a
corresponding id in this sample-->
<?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?>
<data version="3.0">
  <synsets source="dict/data.adv" xml:base="data.adv.xml"><!--10 out of 3621 elements-->
    <synset id="r00001740" type="r">
      <lex_filenum>02</lex_filenum>
      <word lex_id="0">a cappella</word>
      <def>without musical accompaniment</def>
      <example>they performed a cappella</example>
    </synset>
  </synsets>
</data>
```



## 3.2 PRETTIFY:

### 3.2.1 Complexity : $O(n^2)$

The GUI can reorganize the data, it displays it in an organized manner, where each tag is given indents based on its level of hierarchy.

```
<data version="3.0"> <synsets source="dict/data.adv" xml:base="data.adv.xml"> </synsets> <synset
id="r00001740" type="r"> <lex_filenum> 02 </lex_filenum> <word lex_id="0"> a cappella </word> <def> without
musical accompaniment </def> <example> they performed a cappella </example> </synset></synsets></data>
```

The output after hitting the “Prettify Button.”

### 3.2.2 Algorithm Explanation:

We use two algorithms to perform this operation:

1. `scrape_data(string)`:  
Which iterated over the xml file and strips the tags and values into a list.  
Complexity:  $O(n^2)$
2. `prettify_data(list_of_tags)`:  
Which iterates over the list and assigns a power level to each item in the list based on how many parents it has processed before it.  
Complexity:  $O(n)$

```
<data version="3.0">
  <synsets source="dict/data.adv" xml:base="data.adv.xml">
    <synset id="r00001740" type="r">
      <lex_filenum>
        02
      </lex_filenum>
      <word lex_id="0">
        a cappella
      </word>
      <def>
        without musical accompaniment
      </def>
      <example>
        they performed a cappella
      </example>
    </synset>
  </synsets>
</data>
```

## 3.3 CONVERT TO JSON:

### 3.3.1 Complexity : $O(n^2)$

The user has the option to convert the XML file to JSON file.

### 3.3.2 Algorithm Explanation:

- The program uses regular expressions to parse the XML file and creates a dictionary. A python dictionary is a map of key-value pairs where the value can be accessed through its unique key.
- The program then iterates over this dictionary to display it in a JSON-like format.

Example XML input:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A random selection of elements from data.xml Some IDREFS (refs attribute of element
pointer) do not have a corresponding id in this sample-->
<?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?>
<data version="3.0">
  <synsets source="dict/data.adv" xml:base="data.adv.xml"><!--10 out of 3621 elements-->
    <synset id="r00001740" type="r">
      <lex_filenum>02</lex_filenum>
      <word lex_id="0">a cappella</word>
      <def>without musical accompaniment</def>
      <example>they performed a cappella</example>
    </synset>
    <synset id="r00261389" type="r">
      <lex_filenum>02</lex_filenum>
      <word lex_id="0">agonizingly</word>
      <word lex_id="0">excruciatingly</word>
      <word lex_id="0">torturously</word>
      <pointer refs="a01711724" source="3" target="6">Derived from adjective</pointer>
      <pointer refs="a01711724" source="2" target="3">Derived from adjective</pointer>
      <pointer refs="a01711724" source="1" target="1">Derived from adjective</pointer>
      <def>in a very painful manner</def>
      <example>the progress was agonizingly slow</example>
    </synset>
    <synset id="r00423888" type="r">
      <lex_filenum>02</lex_filenum>
      <word lex_id="0">rallentando</word>
      <pointer refs="n07020895">Domain of synset - TOPIC</pointer>
      <def>slowing down</def>
      <example>this passage should be played rallentando</example>
    </synset>
```



## The JSON

e.xml

Style Operations



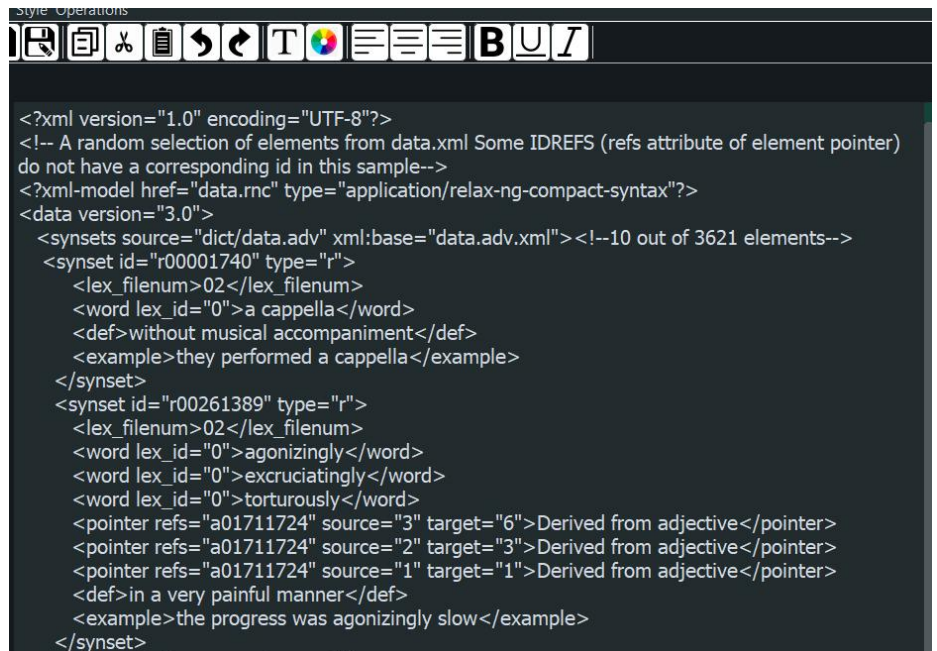
```
{
  "data":
  {
    "synsets":
    {
      "synset":
      {
        "example": "she was a surpassingly beautiful woman"
      }
      "def": "to a surpassing degree"
    }
    "pointer":
    {
      "_text": "Derived from adjective"
      "_refs": "a01676026" _source: "1" _target: "5"
    }
    "word":
    {
      "_text": "surpassingly"
      "_lex_id": "0"
    }
    "lex_filenum": "02"
  }
  "_id": "r00471945" _type: "r"
}
"synset":
{
  "example": "this passage should be played rallentando"
}
"def": "slowing down"
}
"pointer":
{
  "_text": "Domain of synset - TOPIC"
  "_refs": "n07020895"
}
"word":
```

output:

## 3.4 MINIFY:

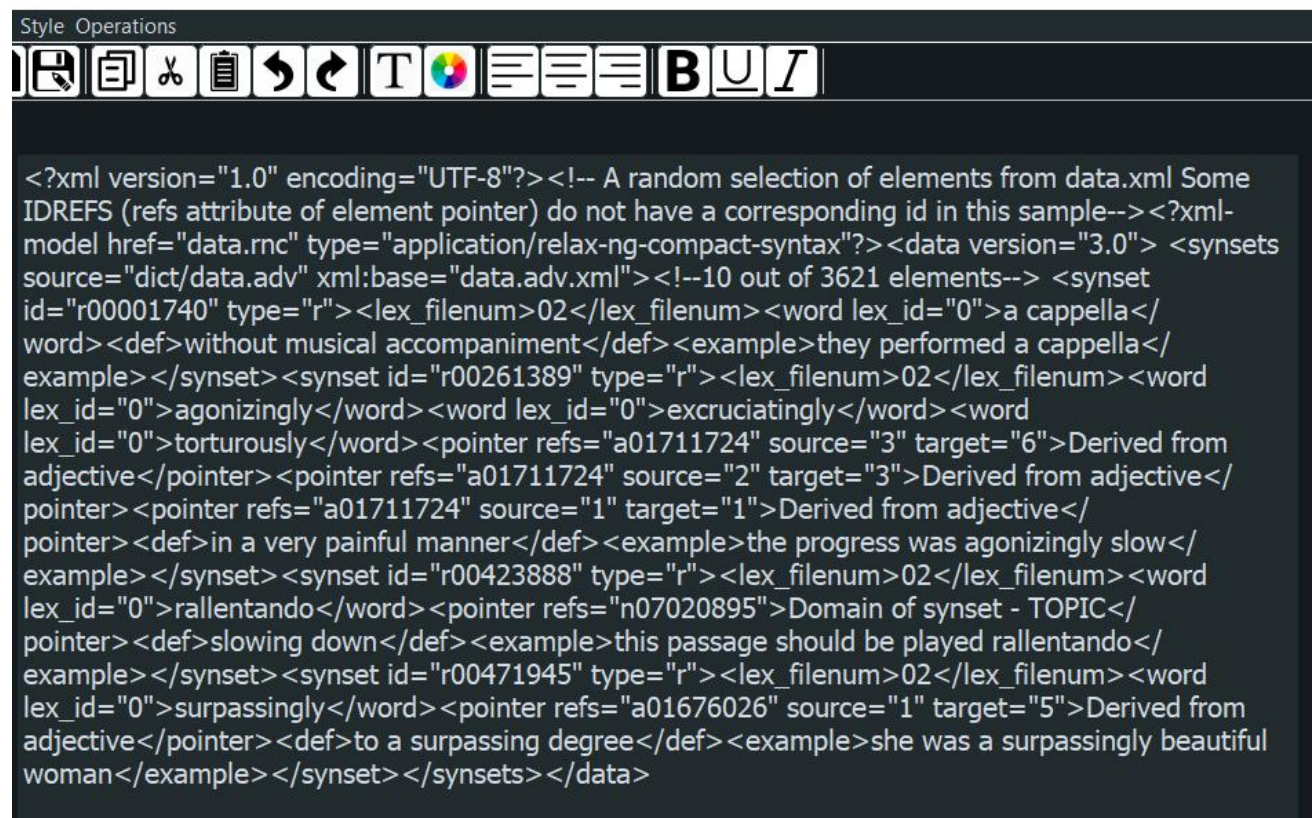
### 3.4.1 Complexity: $O(n)$

By using this option, the program removes the white spaces from everywhere, except inside the tag declaration, as long as the new lines.



```
<?xml version="1.0" encoding="UTF-8"?><!-- A random selection of elements from data.xml Some IDREFS (refs attribute of element pointer) do not have a corresponding id in this sample--><?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?><data version="3.0"><synsets source="dict/data.adv" xml:base="data.adv.xml"><!--10 out of 3621 elements--><synset id="r00001740" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">a cappella</word><def>without musical accompaniment</def><example>they performed a cappella</example></synset><synset id="r00261389" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">agonizingly</word><word lex_id="0">excruciatingly</word><word lex_id="0">torturously</word><pointer refs="a01711724" source="3" target="6">Derived from adjective</pointer><pointer refs="a01711724" source="2" target="3">Derived from adjective</pointer><pointer refs="a01711724" source="1" target="1">Derived from adjective</pointer><def>in a very painful manner</def><example>the progress was agonizingly slow</example></synset>
```

The output of the “Minify” Operation:



```
<?xml version="1.0" encoding="UTF-8"?><!-- A random selection of elements from data.xml Some IDREFS (refs attribute of element pointer) do not have a corresponding id in this sample--><?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?><data version="3.0"><synsets source="dict/data.adv" xml:base="data.adv.xml"><!--10 out of 3621 elements--><synset id="r00001740" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">a cappella</word><def>without musical accompaniment</def><example>they performed a cappella</example></synset><synset id="r00261389" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">agonizingly</word><word lex_id="0">excruciatingly</word><word lex_id="0">torturously</word><pointer refs="a01711724" source="3" target="6">Derived from adjective</pointer><pointer refs="a01711724" source="2" target="3">Derived from adjective</pointer><pointer refs="a01711724" source="1" target="1">Derived from adjective</pointer><def>in a very painful manner</def><example>the progress was agonizingly slow</example></synset><synset id="r00423888" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">rallentando</word><pointer refs="n07020895">Domain of synset - TOPIC</pointer><def>slowing down</def><example>this passage should be played rallentando</example></synset><synset id="r00471945" type="r"><lex_filenum>02</lex_filenum><word lex_id="0">surpassingly</word><pointer refs="a01676026" source="1" target="5">Derived from adjective</pointer><def>to a surpassing degree</def><example>she was a surpassingly beautiful woman</example></synset></synsets></data>
```

## 3.5 COMPRESS:

### 3.5.1 Complexity: $O(n \log n)$

The GUI provides the ability to compress files to reduce their size, then decompresses them to restore the original data.

### 3.5.2 Compression Algorithm:

The compression is done over two steps:

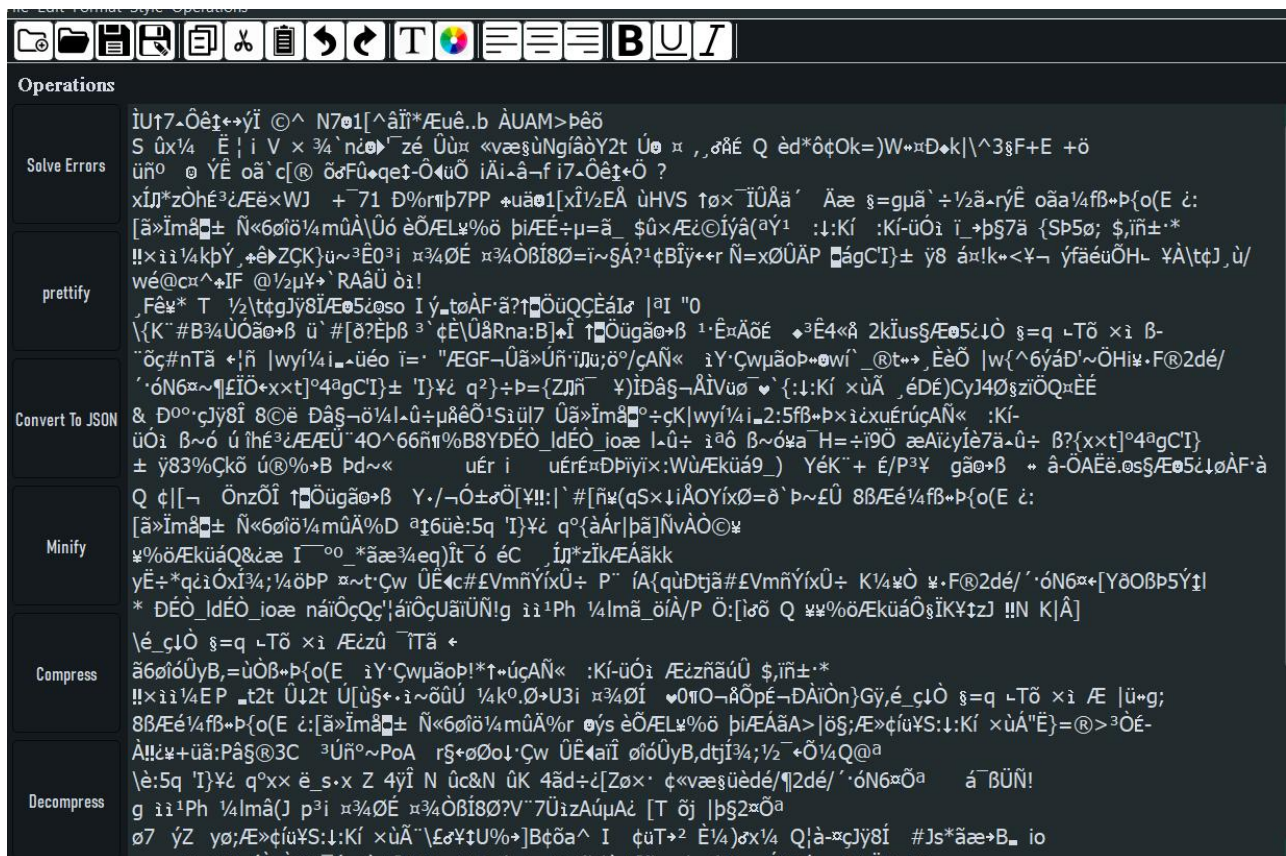
1. The GUI uses Huffman Coding Algorithm for compression. The way this works is that it loops through the file and determines the number of times each character was repeated. It then sorts those repetition numbers in an ascending order. Then it creates a tree from the bottom-up, starting with the characters that was repeated the least to represent the leaves of the tree. After constructing the tree, it assigns each character a binary value based on its position in the tree, where the characters that were repeated the most would get the shortest binary value.
2. We then implemented an encoding algorithm that takes the binary stream output from the Huffman Coding Algorithm. It separates it into 8-bit-chunks and converts each chunk into its equivalent ASCII character, the last few bits have zeros added to their left until they are 8-bits long.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A random selection of elements from data.xml
      Some IDREFS (refs attribute of element pointer) do not have a corresponding id in this sample-->
<?xml-model href="data.rnc" type="application/relax-ng-compact-syntax"?>
<data version="3.0">
  <synsets source="dict/data.adj" xml:base="data.adj.xml"><!--10 out of 18156 elements-->
    <synset id="a00001740" type="a">
      <lex_filenum>00</lex_filenum>
      <word lex_id="0">able</word>
      <pointer refs="n05200169 n05616246">Attribute</pointer>
      <pointer refs="n05616246 n05200169" source="1" target="1">Derivationally related form</pointer>
      <pointer refs="a00002098" source="1" target="1">Antonym</pointer>
      <def>(usually followed by `to') having the necessary means or skill or know-how or authority to do something</
def>
      <example>able to swim</example>
      <example>she was able to program her computer</example>
      <example>we were at last able to buy a car</example>
      <example>able to get a grant for the project</example>
    </synset>
    <synset id="a00327541" type="s">
      <lex_filenum>00</lex_filenum>
      <word lex_id="0">cancellate</word>
      <word lex_id="0">cancellated</word>
      <word lex_id="0">cancellous</word>
      <pointer refs="a00327031">Similar to</pointer>
      <pointer refs="n06057539">Domain of synset - TOPIC</pointer>
      <def>having an open or latticed or porous structure</def>
    </synset>
```





The encrypted output:



The user can save this data and would later have the ability to decompress it.

Comparing the sizes of the before and after the compression:

 data-sample	18 KB
 data-sample-compressed	4 KB

## 3.6 DECOMPRESSION:

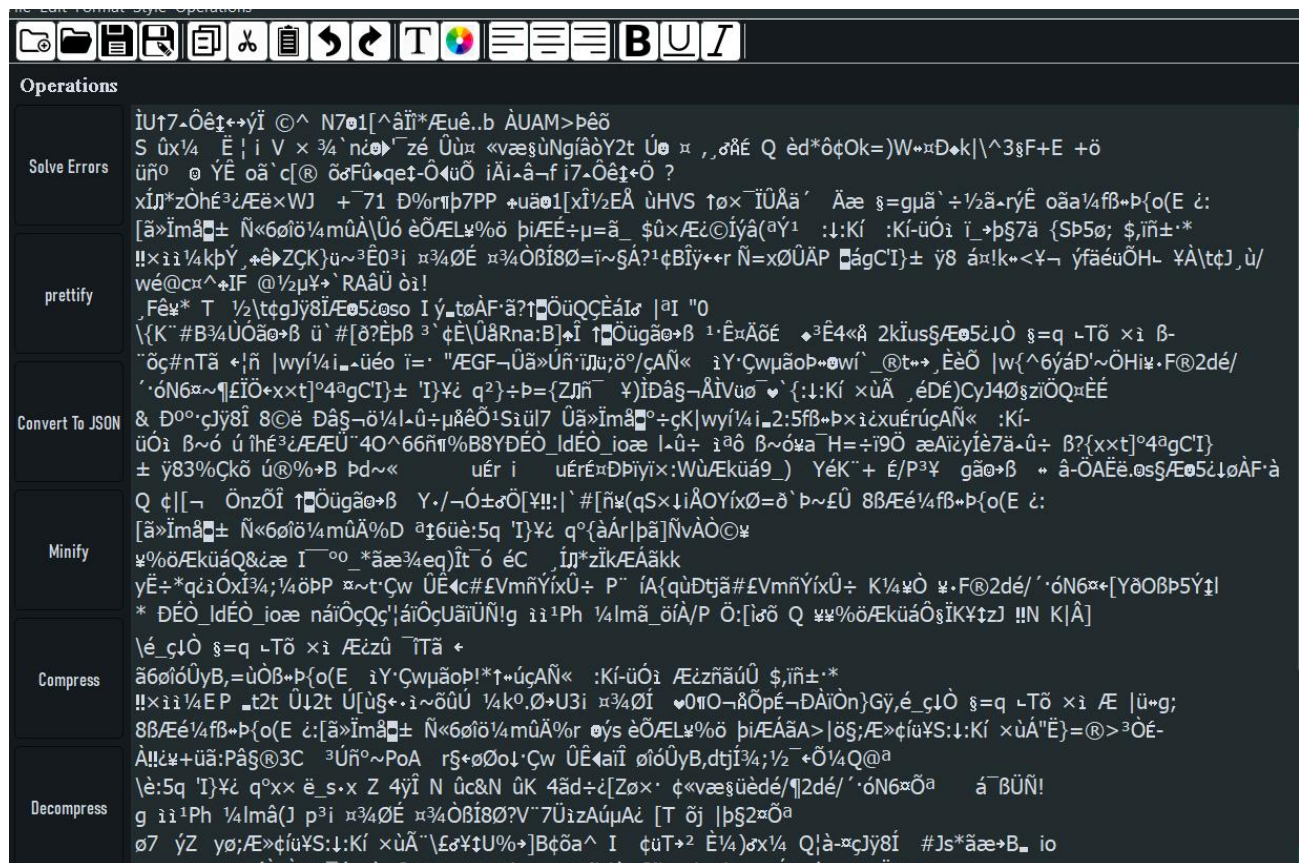
### 3.6.1 Complexity: $O(n)$

The GUI provides the user with the ability to decompress the compressed files, to restore the original data.

The decompression process is done over two steps:

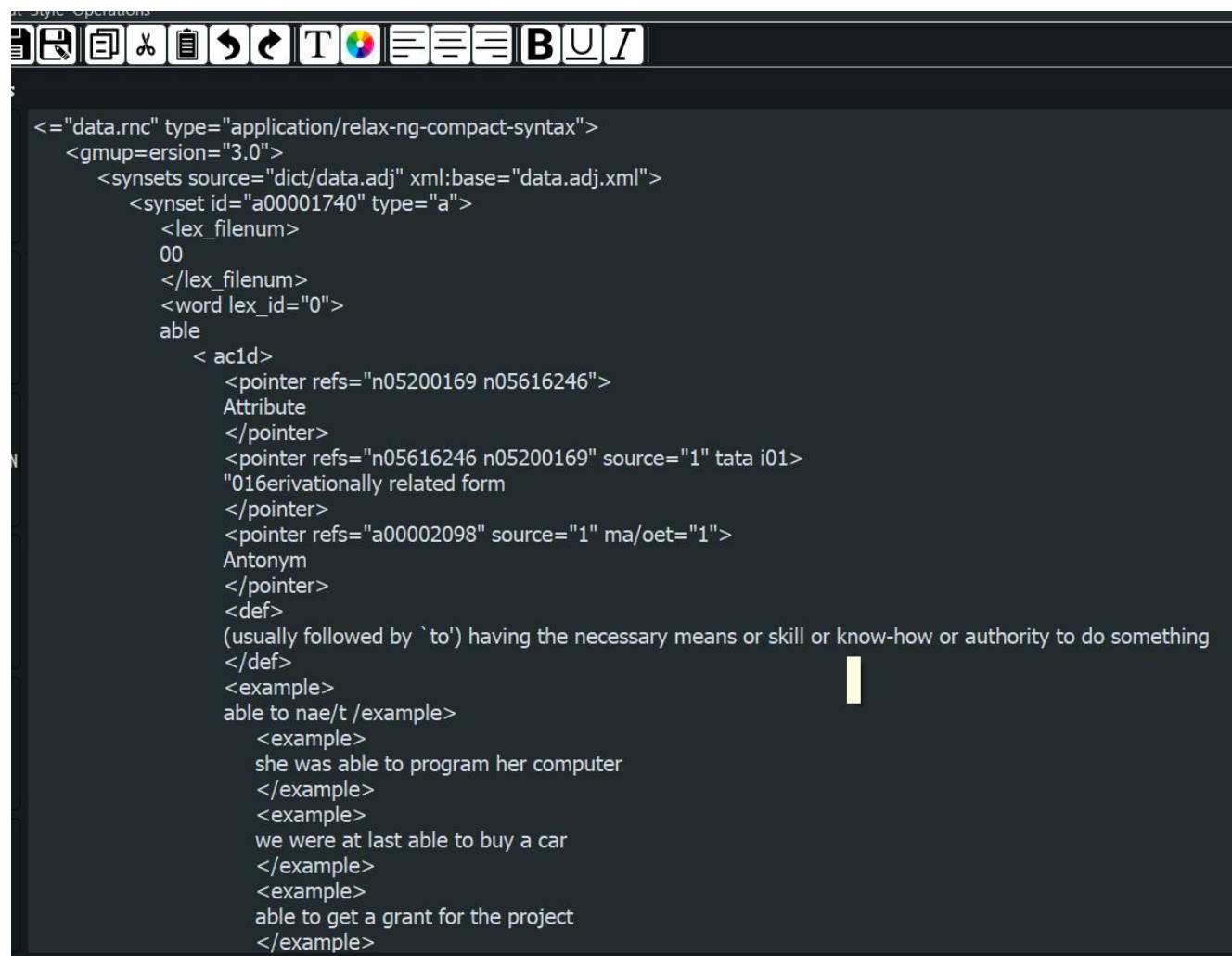
1. Decoding: The GUI takes the encrypted ASCII string from the compressed file and reconstructs its binary form.
2. Binary to string: Takes the bitstream from the decoding function and reconstructs the original data, based on the hash table generated by the Huffman Coding Algorithm in the compression phase.

Example: Decompressing the file that we compressed in the Compress section:








The reconstructed data output:



The screenshot shows a text editor window with a toolbar at the top containing icons for file operations (save, open, copy, paste, undo, redo), text formatting (bold, italic, underline, strikethrough), and alignment. The editor displays XML data for the word 'able'. The XML structure includes a root element with type 'application/relax-ng-compact-syntax', a version attribute, a synset source, and a synset ID. The word 'able' is defined with a pointer to a dictionary entry, a definition, and several examples.

```
<="data.rnc" type="application/relax-ng-compact-syntax">
  <gmup=ersion="3.0">
    <synsets source="dict/data.adj" xml:base="data.adj.xml">
      <synset id="a00001740" type="a">
        <lex_filenum>
          00
        </lex_filenum>
        <word lex_id="0">
          able
          < ac1d>
            <pointer refs="n05200169 n05616246">
              Attribute
            </pointer>
            <pointer refs="n05616246 n05200169" source="1" tata i01>
              "016erivationally related form
            </pointer>
            <pointer refs="a00002098" source="1" ma/oet="1">
              Antonym
            </pointer>
            <def>
              (usually followed by `to') having the necessary means or skill or know-how or authority to do something
            </def>
            <example>
              able to nae/t /example>
              <example>
                she was able to program her computer
              </example>
              <example>
                we were at last able to buy a car
              </example>
              <example>
                able to get a grant for the project
              </example>
```

Comparing the sizes before and after decompression:

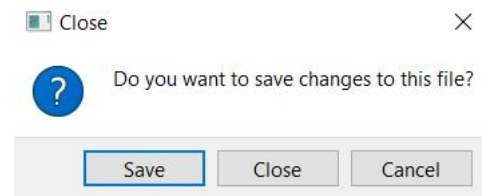
 data-sample	18 KB
 data-sample-compressed	4 KB
 sample-data-reconstructed	18 KB

## 4 GUI ERROR HANDLING:

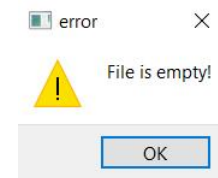
---

The GUI prompts the user with many information messages during runtime, such as:

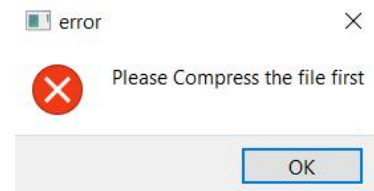
- If the user tries to close the GUI without saving, the GUI asks them if they would like to save before they exit.



- If the user tries to operate on an empty file, the GUI warns them that the file is empty.



- If the user tries to decompress a file that was not compressed by the GUI.



## 5 REFERENCES AND RESOURCES:

---

Some of the resources used during the project:

- W3S Python documentation. <https://www.w3schools.com/python/>
- Python documentation. <https://docs.python.org/3/>
- PyQt5 documentation. <https://doc.qt.io/qtforpython/>
- Article: The Absolute Minimum Every Software Engineer need to understand about encoding and Unicode (OCTOBER 8, 2003 by JOEL SPOLSKY.) <https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/>
- XML Validator. <https://codebeautify.org/xmlvalidator>  
<https://www.xmlvalidation.com/>
- XML to JSON converter. <https://www.freeformatter.com/xml-to-json-converter.html>

## 6 LINK TO THE GITHUB REPOSITORY:

---

[https://github.com/NevenNabil/XML\\_Editor](https://github.com/NevenNabil/XML_Editor)

## 7 LINK TO THE YOUTUBE VIDEO:

---

<https://www.youtube.com/watch?v=WvKDkc76ku8>