A blue and black logo with birds and text

Description automatically generated

**CSE331: Data Structure and Algorithms**

**Project report**

|  |  |  |
| --- | --- | --- |
| Name | Id | Contribution |
| Moustafa Ahmed | **2100467** | leader/GUI/CLI/draw graph |
| Kareem Ehab | **2100913** | GUI/UX |
| Michael George | **2100709** | Minify/compress/decompress |
| Moaz Ragab | **2100938** | validation correction formatting |
| Ahmed Ashraf | **2100255** |
| Karen Maurice | **2100748** | Xml to Jason/report |
| Mohamed Waleed | **2100623** | Graph/social network graph |
| Mayar Walid | **2100953** | Most active/influencer  mutual follower |
| Mohamed Hani | **2100915** | search |
| Amr Essam | **2001089** | User’s suggestions |

[***GitHub***](https://github.com/MoustafaHashem/XMLEditorProject)

**XML Editor and Social Network Visualizer**

**Table of Contents:**

Table of Contents

[**1. Background** 3](#_Toc185456067)

[**2. Implementation Details** 3](#_Toc185456068)

[**3. Complexity of Operations** 6](#_Toc185456069)

[**4. Conclusion** 7](#_Toc185456070)

[**5. References** 7](#_Toc185456071)

**1. Background**

**1.1 Introduction to XML**

Extensible Markup Language (XML) is a versatile markup language designed to store and transport data in a structured format. It is widely used in various applications for data interchange due to its ability to represent complex data structures in a human-readable format. However, working with XML files often requires specialized tools for parsing, analyzing, and visualizing data.

**1.2 Project Overview**

This project focuses on developing a desktop application that facilitates the parsing and visualization of XML files and many other operations like formatting, compression/decompression and conversion to JSON. Also, it represents user data in a social network. Each user within the XML file has attributes such as a unique ID, name, a list of posts, and followers. The application aims to provide various functionalities, enabling users to manipulate and analyze this data effectively.

**1.3 Key Features**

The application offers a comprehensive set of features that include:

* **XML Parsing and Verification:** Validates the structure of XML documents to ensure data integrity.
* **XML Formatting:** Prettifies XML files to improve readability.
* **XML to JSON Conversion:** Converts XML data into JSON format for compatibility with JavaScript applications.
* **File Minification:** Reduces file size by removing unnecessary whitespace and indentations.
* **Data Compression:** Compresses XML/JSON files using Byte Bair Encoding Technique.
* **Graphical Representation of User Relationships:** Visualizes user connections using graph data structures.
* **Network Analysis Tools:** Identifies influential users, mutual followers, and suggests connections.
* **Post Searching Capabilities:** Allows users to search posts by keywords or topics.

This application caters to both casual users through a GUI and advanced users via a command-line interface (CLI), enhancing accessibility and usability.

**2. Implementation Details**

**2.1 General Architecture**

The application consists of the following components:

1. **XML Processor:** Handles parsing, verifying, formatting, and minifying XML files.
2. **Data Converter:** Converts XML to JSON and vice versa.
3. **File Compressor:** Compresses and decompresses files using a custom or standard compression technique.
4. **Graph Module:** Builds and visualizes user relationships using a graph data structure.
5. **Network Analyzer:** Extracts insights such as the most influential user, the most active user, mutual followers, and follow suggestions.
6. **Search Module:** Allows searching posts by keywords or topics.

**2.2 Graphical User Interface (GUI)**

The GUI is designed for user-friendliness and supports:

* Browsing input files.
* Displaying outputs in a read-only text field.
* Saving output files to user-defined locations.
* Buttons for each supported operation.

**2.3 Command-Line Interface (CLI)**

The CLI allows advanced users to execute operations through commands like:

*xml\_editor verify -i input\_file.xml -o output\_file.xml*

*xml\_editor json -i input\_file.xml -o output\_file.json*

Levenshtein Distance algorithm to correct wrong commands (getClosestCommand())

**2.4 Functionalities**

**2.4.1 XML Consistency Check**

The application validates XML structure by:

* Ensuring all opening tags have corresponding closing tags.
* Highlighting errors and their line numbers in case of inconsistencies.
* Offering an option to auto-correct errors and save the fixed file.

**2.4.2 Formatting (Prettifying)**

Improves XML readability by:

* Indenting nested elements.
* Removing unnecessary whitespace while maintaining structure.

**2.4.3 XML to JSON Conversion**

Converts XML data to JSON format for easier integration with JavaScript-based tools.

**2.4.4 Minifying XML**

Reduces file size by:

* Removing unnecessary whitespace and newlines.

**2.4.5 Compression and Decompression**

[Byte Bait Encoding](https://en.wikipedia.org/wiki/Byte_pair_encoding) is the algorithm used to compress any kind of text file. After compression, the program saves the key pairs of the encoding process followed by the encoded data.

Concerning the decompression operation, the program reads the file ready to fetch the pairs to be able to apply Byte Pair Decoding on the text that follows.

**2.4.6 Graph Representation**

Represents user relationships as a directed graph:

* Nodes represent users.
* Edges represent follower relationships.

**2.4.7 Network Analysis**

Analyzes the social network to:

* Identify the most influential user (maximum followers).
* Identify the most active user (maximum connections).
* Find mutual followers for a set of users.
* Suggest users to follow (followers of followers).

**2.4.8 Post Search**

Enables searching for posts containing specific keywords or topics.

**3. Complexity of Operations**

**3.1 XML Parsing and Validation**

* **Time Complexity:**

O(n) where n is the total number of tokens in all lines.

* **Space Complexity:**

O(n⋅m) where n is the number of lines, m is the average length of a line.

**3.2 Formatting (Prettifying)**

* **Time Complexity:**

O(n.m) where n is number of lines and m is the average length of line in words

* **Space Complexity:**

O(n.m)

**3.3 XML to JSON Conversion**

* **Time Complexity:** O(n⋅m+L+N), where: n is the number of XML lines, m is the average length of each XML line, L is the total length of the XML string, N is the total number of elements in the resulting JSON.
* **Space Complexity:** O(L+N), where: L is the length of the XML string and the formatted JSON string, N is the number of elements in the JSON structure.

**3.4 Graph Representation**

* **Time Complexity:**
  + addVertex: O(1) — Constant time for HashMap's putIfAbsent.
  + buildGraphFromXML: O(n) — Processes each character in the XML string.
  + getAdjacencyList: O(1) — Constant time for HashMap lookup.
  + areConnected: O(d) — Searches adjacency list, where d is the number of followers.
  + isFollowing: O(d) — Searches adjacency list, where d is the number of followers.
* **Space Complexity:**
  + Adjacency List: O(V + E) — *V* vertices (users) and *E* edges (relationships).
  + User Objects: O(V) — Stores user data.
  + buildGraphFromXML: O(n) — XML processing.
  + Pattern Matching: O(1) — Constant space for pattern and matcher.
  + Posts and Followers: O(P + F) — *P* posts and *F* followers.
  + Overall: O(V + E + P + F).

**3.5 Network Analysis**

* **Most Influential/Active User:**
  + **Time Complexity:**
* Most Influential:
* ReadFile(s): O(F).
* Both for loops: O(n).
* Overall complexity of function is O(F) + O(n) = O(F + n).
* Active User:
* ReadFile(s): O(F)
* Outer and inner loops: O(n^2).
* Another loop: O(n).
* Combining the above, the overall complexity is O(F + n^2)
  + **Space Complexity:**
* Most influential:
* Space for users: O(n).
* Space for mostInf: O(n).
* Overall, the space complexity is O(n).
* Active User:
* Space for users and mostActive: O(n) space.
* Graph: Depends on the graph representation O(n+e) where e is the number of edges (connections).
* Overall, the space complexity is O(n + e).
* **Mutual Followers:**
  + **Time Complexity:**
    - ReadFile(s): O(F).
    - Outer loop: O(n).
    - Inner loops: O(m).
    - Overall, the complexity is O(F)+O(n⋅m).
  + **Space Complexity:**
    - Space for user and mutalIF: The users array and mutualF list require O(n) space.
    - Graph: Depends on the graph representation: O(n+e) where e is the number of edges (connections).
    - Overall, the space complexity is O(n + e).
* **Follow Suggestions:**
  + **Time Complexity:** O(f\*k), worst-case O(n^2).
  + **Space Complexity:** O(f\*k), worst-case O(n^2).

**3.6 Compression and Decompression**

* **Minifying:**
  + **Time Complexity:** O (N)
  + **Space Complexity:** O (N)
* **Compression:**
  + **Time Complexity:** O (N + t ×N + Lkeys + Lvalues)
  + **Space Complexity:** O (N + Lkeys + Lvalues)

t represents the number of iterations of the outer while loop.

* **Decompression:**
  + **Time Complexity:** O (M + k x N + N)
  + **Space Complexity:** O(k + N)
    - M: Length of the hashmap string s[0].
    - k: Number of pairs in the hashmap.
    - N: Length of the encoded data s[1].

**3.7 Seach**

* **Word search:**
  + **Time Complexity:**
  + **Space Complexity:**
* **Topic search:**
  + **Time Complexity:**
  + **Space Complexity:**

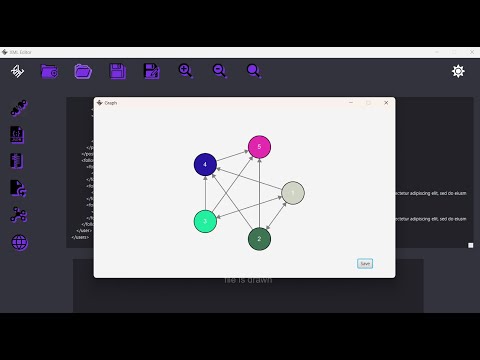
**4. Conclusion**

This application provides a robust solution for parsing, analyzing, and visualizing XML data representing social networks. The combination of GUI and CLI modes ensures accessibility for both casual and advanced users. By integrating network analysis and search functionalities, it goes beyond basic XML operations, offering insights into user relationships and activity.

**5. References**

* [**Byte pair encoding - Wikipedia**](https://en.wikipedia.org/wiki/Byte_pair_encoding)
* [**Byte-Pair Encoding tokenization - Hugging Face NLP Course**](https://huggingface.co/learn/nlp-course/en/chapter6/5)

**6. Video**

**[](https://www.youtube.com/embed/avsl-Ycp0vg?feature=oembed)**

[***GitHub***](https://github.com/MoustafaHashem/XMLEditorProject)