Software Requirements Specification (SRS)

Project Title: Advanced Text Editor
Project members: Ayesha imran (2023-bsai-011)--Leader
Hassan Tariq (2023-bsai-004)
Qandeel Asim (2023-bsai-037)
Taibah Shehbaz (2023-bsai-024)
Submitted to : Mam Irsha Qureshi
Date: 14-Janurary-2025

---

1. Introduction

1.1 Purpose

The purpose of this project is to develop an Advanced Text Editor application that provides fundamental text editing functionalities such as adding, deleting, copying, and pasting lines, along with advanced features like undo/redo, search and replace, and clipboard support. This tool is designed for users who require lightweight and efficient text management in a command-line interface.

1.2 Scope

This Advanced Text Editor will be implemented using C++ and will provide users with a menu-driven interface to manage text documents efficiently. The project is targeted at students and professionals seeking a basic, console-based text editing application.

1.3 Definitions, Acronyms, and Abbreviations

- **CLI:** Command-Line Interface
- **Undo/Redo:** Features that allow reverting or reapplying the last performed actions
- **Clipboard:** Temporary storage for copying and pasting text

1.4 References

- C++ Standard Library Documentation
- Basic Data Structures and Algorithms Textbook

---

2. System Overview

The Advanced Text Editor application operates in a CLI environment and allows users to interact with a document through predefined menu options. It includes core functionalities like text management and search/replace operations.

---

3. Functional Requirements

3.1 Add Line

- **Description:** Allow the user to append a new line to the document.
- **Input:** Text string to be added.
- **Output:** Updated document with the new line.

3.2 Delete Line

- **Description:** Remove the last line from the document.
- **Input:** None.
- **Output:** Updated document with the last line removed.

3.3 Copy Line

- **Description:** Copy a specified line to the clipboard.
- **Input:** Line number.
- **Output:** Line content stored in the clipboard.

3.4 Paste Line

- **Description:** Paste the clipboard content as a new line.
- **Input:** None.
- **Output:** Updated document with the pasted line.

3.5 Undo

- **Description:** Revert the last change made to the document.
- **Input:** None.
- **Output:** Document state prior to the last change.

3.6 Redo

Description: Reapply the last undone action.
Input: None.
Output: Document state after reapplying the last undone action.

3.7 Search Word

- **Description:** Find and display lines containing a specific word.
- **Input:** Word to search.
- **Output:** List of line numbers and content where the word is found.

3.8 Replace Word

- **Description:** Replace occurrences of a word in the document with a new word.
- **Input:** Old word and new word.
- **Output:** Updated document with the word replaced.

3.9 Display Document

- **Description:** Display the entire content of the document.
- **Input:** None.
- **Output:** List of lines in the document.

3.10 Exit

- **Description:** Terminate the application.
- **Input:** None.
- **Output:** Exit the program with a goodbye message.

---

4. Non-Functional Requirements

4.1 Performance

The editor should process text operations within 1 second for a document of up to 100 lines.

4.2 Usability

Provide clear instructions and feedback for all menu options.

4.3 Reliability

Ensure no data corruption during undo/redo operations.

4.4 Portability

The application should run on any platform supporting a C++ compiler.

4.5 Maintainability

Code should be modular and well-documented for easy maintenance and extension.

---

5. System Design

5.1 Data Structures

- **2D Array:** To store document lines.
- **Clipboard:** To store the copied line.
- **Undo/Redo Stacks:** To store states for undo/redo operations.

5.2 Algorithms

- **Search Algorithm:** For substring matching.
- **Stack Operations:** For managing undo/redo functionality.

5.3 User Interface

Command-line menu-driven interface with numbered options.

---

6. Constraints

- Maximum of 100 lines in a document.
- Each line can have up to 100 characters.
- Single clipboard storage.

---

7. Testing

7.1 Unit Testing

Test each function individually (e.g., addLine, deleteLine).

7.2 Integration Testing

Test the interaction between multiple features (e.g., undo after addLine).

7.3 User Testing

Conduct tests to ensure the interface is intuitive and operations work as expected.

---

8. Appendices

A. Development Tools

- **Compiler:** GCC 11.3 or higher
- **IDE:** Embarcadero Dev-C++