1. Introduction

The Stack-Based Text Editor is a command-line Python application that enables users to create, edit, and manage text files dynamically. This project incorporates core functionalities of a typical text editor, including file creation, text insertion, deletion, undo, and redo actions. By leveraging stack-based operations, it efficiently handles complex editing actions, allowing for easy management of undo and redo functionalities.

The project is built to be user-friendly and extendable, with features including:

- File management: Users can create new files, load existing ones, and save changes.
- Text editing: Provides basic text insertion and deletion, with automatic space insertion for readability.
- Undo/Redo: The stack-based approach allows for tracking each action, making it easy to reverse or repeat recent changes.
- Dynamic interaction: The command-line interface supports a range of commands for flexible, user-directed editing.

2. <u>Software and Hardware requirements:</u>

Software requirements:

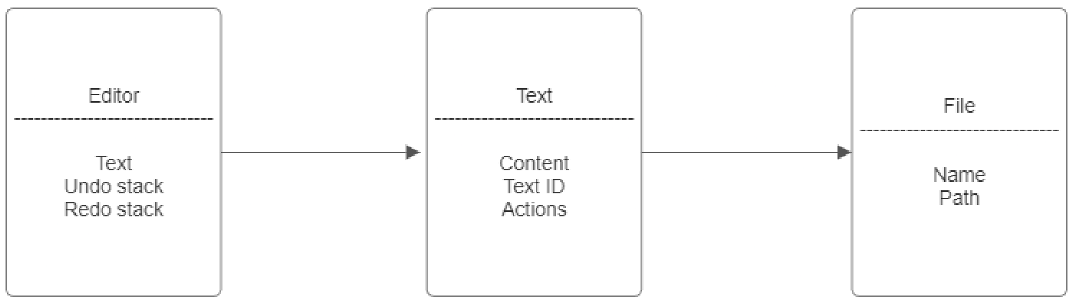
- Operating system: Any OS supporting Python (Windows, macOS, Linux)
- Programming language: Python 3.7 or above
- IDE/Text editor: Any IDE that supports Python, such as Visual Studio Code, PyCharm, or Jupyter Notebook
- Python libraries: No external libraries required

Hardware requirements:

- Processor: Minimum Dual-core 1.3 GHz or higher
- RAM: Minimum 2 GB
- Storage: Minimum 100 MB free space

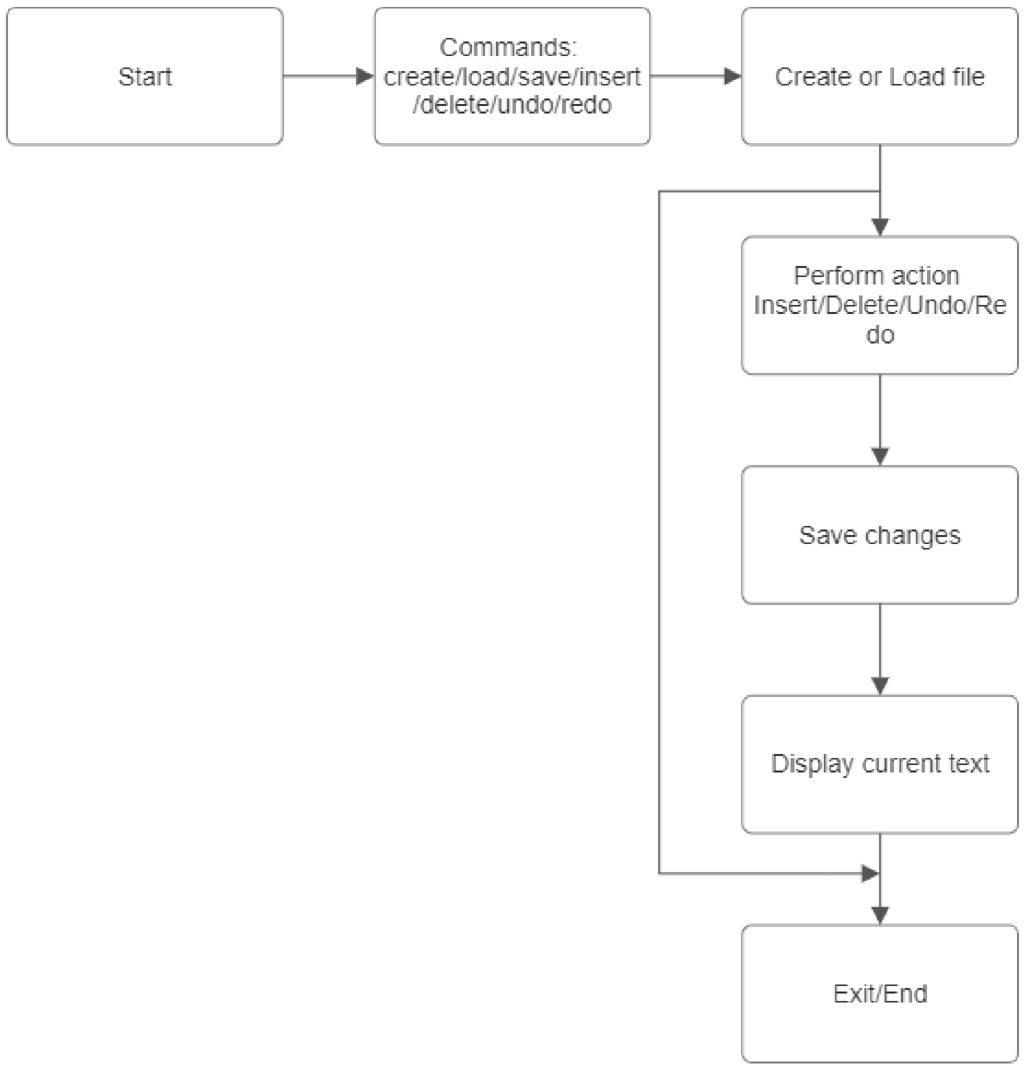
3. ER-Diagram:

The following ER diagram outlines the relationship between the main entities in the Stack-Based Text Editor:



4. Program flow diagram:

The following flow diagram outlines the key program processes:


Start
Commands: create/load/save/insert/delete/undo/redo
Create or Load file
Perform action Insert/Delete/Undo/Redo
Save changes
Display current text
Exit/End

5. <u>Functional component:</u>

1. <u>File management module:</u>
    - Handles file creation, loading, and saving.
    - Sets the current file and updates the text editor's state.
2. <u>Text editing module:</u>
    - Allows users to insert and delete text in the editor.
    - Provides automatic space insertion for improved readability.
3. <u>Undo/Redo module:</u>
    - Uses a stack to manage text changes and supports undoing and redoing recent actions.
4. <u>Command-line interface:</u>
    - Offers an interactive experience with commands for each functionality.
    - Accepts inputs for file management, text manipulation, and navigation commands.

6. Conclusion:

The Dynamic Stack-Based Text Editor effectively demonstrates the use of stack data structures to manage undo and redo operations, providing a robust and efficient mechanism for editing text. By implementing file handling and basic text manipulation features, it mimics a simple text editor while ensuring a seamless user experience. The modular design and intuitive command structure make the system easy to understand and extend. This project serves as a practical example of applying algorithms and data structures to solve real-world problems efficiently.