**Data Structures**

**Final Project**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll No** | **Section** |
| **M.Soban** | **22F-3163** | **BAI-3B** |
| **Areeb Dastgeer** | **22F-3099** | **BAI-3B** |

**PRESENTED TO:**

**Sir Muhammad Adeel**

**National University of Computer and Emerging Sciences**

****

**PROBLEM STATEMENT:**

This project aims to create a C++ program for a Dictionary with essential features such as loading, adding, searching, deleting, updating word’s meaning, and providing word suggestions. The program should mimic the functionalities of a dictionary, allowing users to interact with a collection of words and their meanings.

**DESCRIPTION:**

The C++ Dictionary program is designed to provide users with a complete tool for managing and interacting with a dictionary. The primary features of the program include loading an initial dictionary from a file, adding new words, searching for words and their meanings, deleting words, updating word meanings, and receiving word suggestions based on user input. It also includes a feature for saving their own changes in the desired file.

**PROCEDURE:**

1. **Loading Dictionary:**

Implemented a function to load words and meanings from a text file (Dictionary.txt).

Displayed a “loaded” popup after the dictionary loading process.

1. **Adding Word:**

Implemented functionality to add new words and meanings to the dictionary.

Checked for existing words and displayed a popup if the word already exists.

1. **Searching Word:**

Provided a search option for users to look up word meanings.

Displayed the meaning if the word was found; otherwise, showed a "No Meaning" message.

1. **Deleting Word:**

Implemented a "Delete" button on the search page for removing words.

But the changes are not made directly in the main Dictionary.txt file.

1. **Updating Word:**

Added an "Update" button on the search page to modify word meanings.

Displayed a confirmation popup before updating the word.

1. **Word Suggestion:**

Provided a "Suggest" option on the search page to offer word suggestions.

Displayed up to 10 relevant word suggestions based on user input.

1. **Saving Dictionary:**

Provided a “Save Dictionary” option that saves the changes made, in a new file.

**Trie Tree Implementation:**

The program's underlying data structure for word storage is a Trie Tree. A Trie Tree is particularly efficient for storing and searching for strings. In the context of a dictionary, a Trie allows for quick and efficient retrieval of word meanings.

**Reasons for Implementing Trie Tree:**

1. **Fast Search Time**

Trie Trees have a time complexity of O(L), where L is the length of the searched word.

1. **Prefix Matching:**

Trie Trees naturally support prefix matching. This feature is utilized in the word suggestion functionality.

1. **Easy Insertion and Deletion:**

Inserting and deleting words in a Trie Tree is relatively simple and can be done in O(L) time.

**Comparison with Hashing:**

Choosing a Trie Tree over a hashing approach for the dictionary brings several benefits. Trie Trees are effective for quick searching and easy updates, making them a great fit for dictionaries. Unlike hashing, Trie Trees provide a straightforward path to each word, resulting in fast searches with a complexity of O(L), where L is the word length. This ensures speedy and accurate word retrieval. Moreover, Trie Trees naturally handle word prefixes, making it easier to suggest words based on user input.

In contrast, hashing might face challenges like collisions, potentially leading to slower searches or requiring additional handling. Trie Trees, with their inherent structure and simplicity in adding or removing words, seems to be a better choice for efficiently managing words in a dynamic dictionary.

**Functions Used:**

**insertWord:**

Inserts a new word and its meaning into the dictionary.

**insertWordDic:**

Inserts a new word and meaning, returning a success flag.

**searchWordMeaning:**

Searches for the meaning of a word in the dictionary.

**updateWordMeaning**:

Updates the meaning of an existing word.

**deleteWord**:

Deletes a word from the dictionary.

**suggestWords**:

Provides word suggestions based on a user-entered prefix.

**loadDictionary**:

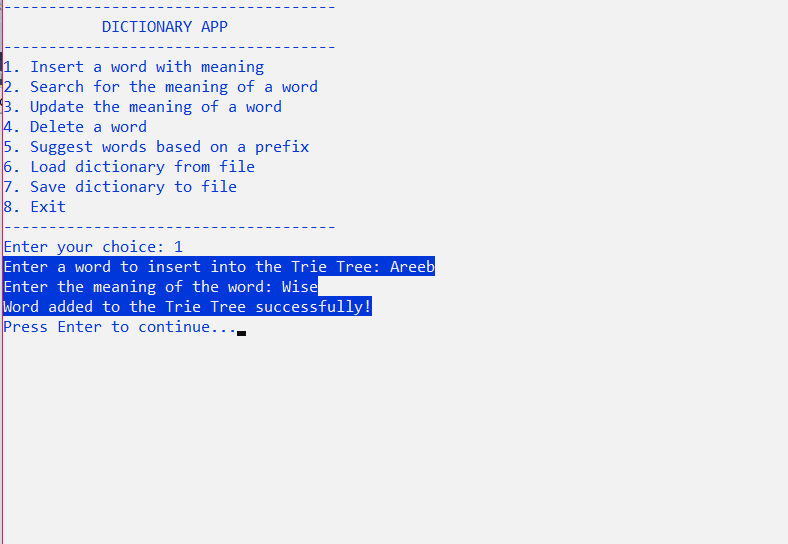
Loads the initial dictionary from a text file.

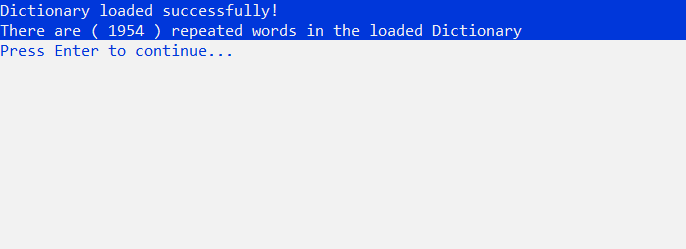
**saveDictionary**:

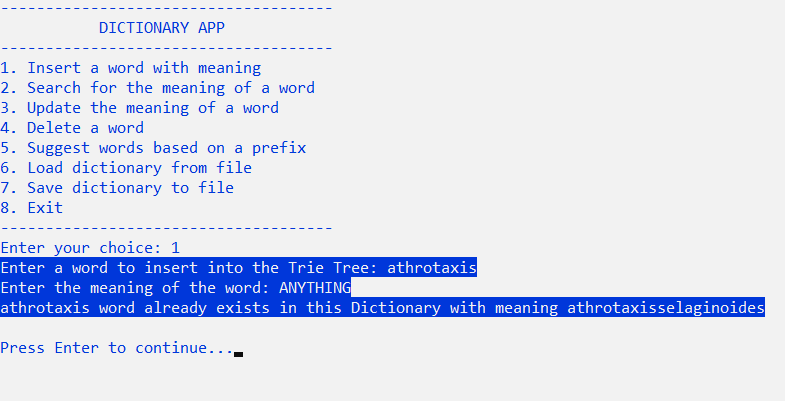
Saves the current dictionary state to a file.

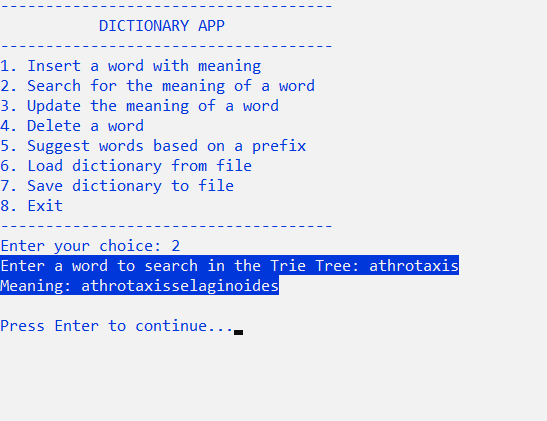
**RESULTS:**

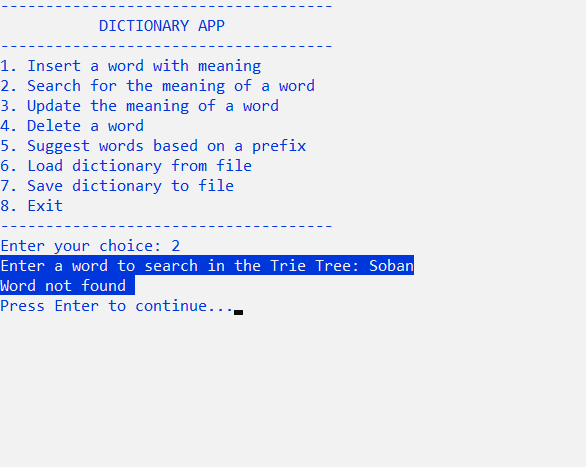
The C++ Dictionary program successfully meets the specified requirements and provides a user-friendly console interface for managing words and their meanings. The incorporation of a Trie Tree data structure enhances the program's efficiency, making it a good solution for a dictionary application.

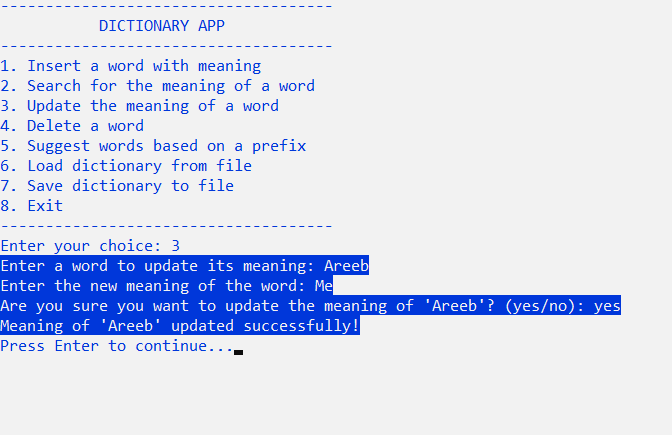


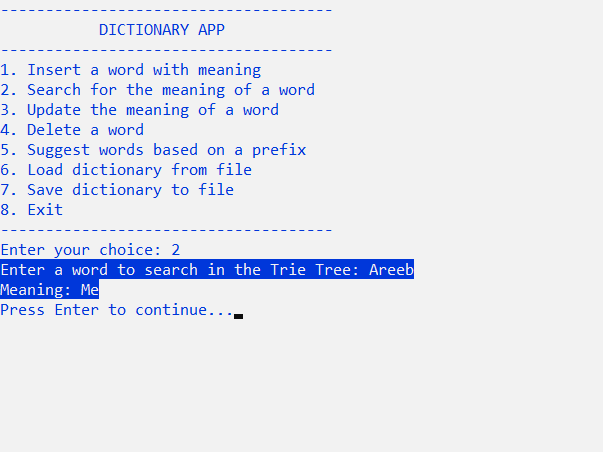


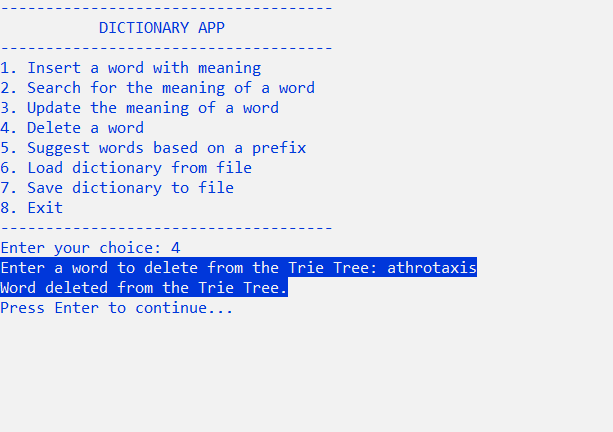


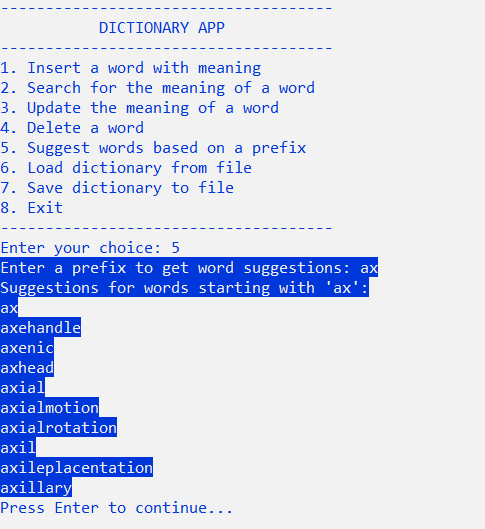
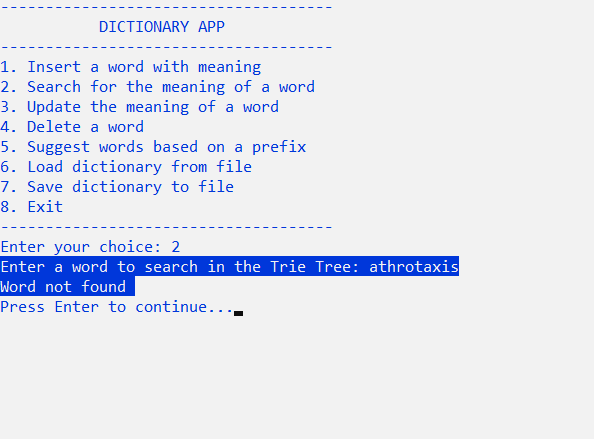


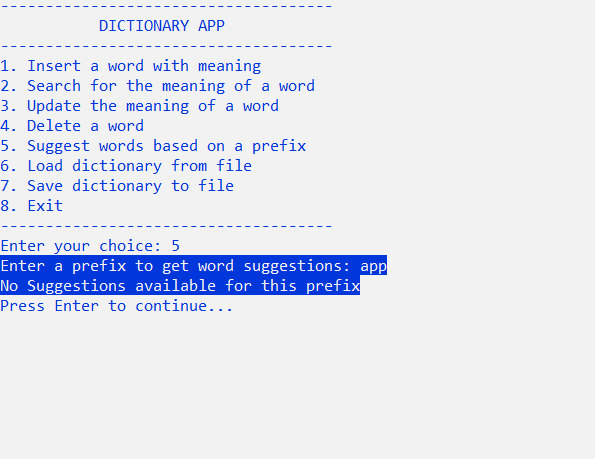


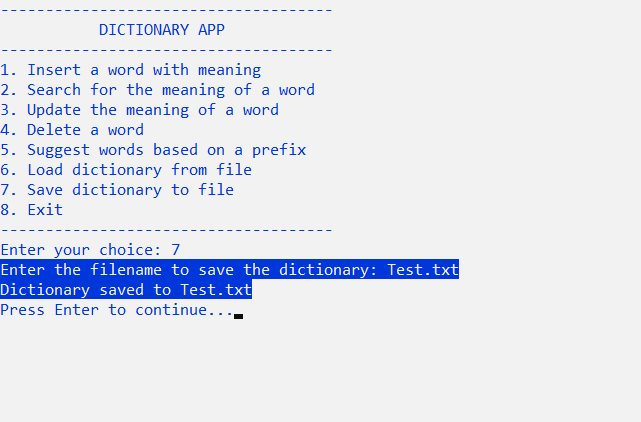


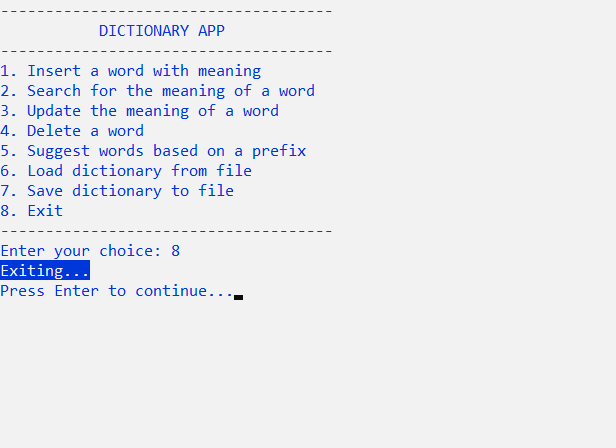


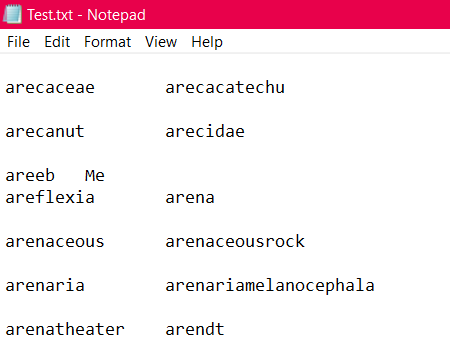












**CONCLUSION:**

In conclusion, the C++ Dictionary project successfully implements essential dictionary functionalities using the data structure Trie Tree . The program delivers a user-friendly console interface, facilitating word management.

Throughout the project, valuable insights were gained in areas such as the utilization of data structures like Trie Trees, and the effective file handling. Challenges encountered during development were met with effective solutions, ultimately playing a positive role in the project's overall success.