Bahria University,

Karachi Campus

COURSE: CSC-221 DATA STRUCTURES AND ALGORITHM
TERM: FALL 2023, CLASS: BSE- 3 (C)

PROJECT NAME

Dictionary

GROUP MEMBERS LIST:

Laiba Shakeel [02-131222-123] (TEAM LEAD)
Hafsa Shahid [02-131222-088]
Afza Mehak Ansari [02-131222-006]

Engr. Majid Kaleem/ Engr. Ayesha Khan

Signed **Remarks:** **Score:**

Table of Contents:

1. INTRODUCTION & PROBLEM:

Language is the heartbeat of human connection, an ever-evolving tapestry of words that weave our thoughts, emotions, and stories. At the heart of this linguistic symphony stands the dictionary – a trusted companion on the journey of comprehension, learning, and linguistic exploration. In our digital era, where convenience is key, the Dictionary project emerges as a beacon, seamlessly blending the traditional wisdom of word repositories with the dynamism of technology. Crafted with care in Java, this application transcends the static nature of conventional dictionaries, inviting users into a realm of versatility and interaction. It is more than a mere lexicon; it is an interactive platform where users can delve into the intricacies of language, add their unique words, update meanings, and curate a dictionary that resonates with their individuality. At its core lies the Binary Search Tree (BST) algorithm, a sophisticated mechanism that ensures not only swift access to words but also efficient management of the lexical landscape. The Dictionary project, with its user-friendly interface and the power of BST, not only bridges the gap between the traditional and the contemporary but also empowers users to curate and enhance their linguistic realms with ease. Welcome to a world where words come alive, where the Dictionary is not just a book on a shelf but a dynamic, digital companion that evolves with you on your journey through language.

2. PARADIGMS:

The project is implemented in Java and utilizes object-oriented programming (OOP) principles. It employs a hierarchical tree structure (BST) for efficient word retrieval and modification operations. File handling is incorporated for loading and updating the dictionary from an external file ("Dic.txt"). Additionally, the program utilizes user interaction through a console-based menu system.

3. ALGORITHM & EXPLANATION:

1. Constructor and Destructor

- **Dictionary():** Initializes a new Dictionary object.
- **~Dictionary():** Destructor to clean up memory used by the Dictionary object.

2. File Operations

- **LoadDictionary():** Loads the dictionary from a file ("Dic.txt") and constructs the tree.
- **UpdateFile():** Updates the file ("Dic.txt") with the current state of the dictionary.
- ***WriteToFile(ofstream& outFile, DictionaryNode node)**:** Helper function to recursively write the dictionary to a file.

3. Tree Operations

- **Insert(const string& word, const string& meaning):** Inserts a new word into the dictionary.
- ***Insert(DictionaryNode node, const string& word, const string& meaning)**:** Helper function for inserting a new word recursively.
- ***Search(DictionaryNode node, const string& word)**:** Searches for a word in the dictionary.
- ***FindMin(DictionaryNode node)**:** Finds the minimum node in a subtree.
- ***Delete(DictionaryNode node, const string& word)**:** Deletes a word from the dictionary.

4. Dictionary Operations

- **AddWord(const string& word, const string& meaning):** Adds a new word to the dictionary if it doesn't already exist.
- **SearchWord(const string& word):** Searches for a word in the dictionary.
- **DeleteWord(const string& word):** Initiates the process of deleting a word from the dictionary.
- **UpdateWord(const string& word, const string& newMeaning):** Initiates the process of updating the meaning of a word.

- **WordSuggestion(const string& partialWord):** Provides suggestions for words based on a partial input.

5. Printing

- **InOrderPrint():** Initiates the process of printing the dictionary in an in-order traversal.
- ***InOrderPrint(DictionaryNode node)**:** Helper function to recursively print the dictionary in an in-order traversal.

6. Time Complexity

The overall time complexity is influenced by the individual operations within the loop in the main function. The approximate overall time complexity is O(T * (n log n + nm)), where T is the number of iterations (user interactions), n is the number of nodes in the BST, log n represents the logarithmic complexity of typical BST operations, and m is the length of the partial word.

4. ALGORITHM CODE:

```
public void Insert(String word, String meaning) {
    root = Insert(root, word, meaning);
}

public DictionaryNode Insert(DictionaryNode node, String word, String meaning) {
    if (node == null) {
        return new DictionaryNode(word, meaning);
    }
    if (word.compareTo(node.word) < 0) {
        node.left = Insert(node.left, word, meaning);
    } else if (word.compareTo(node.word) > 0) {
        node.right = Insert(node.right, word, meaning);
    }
    return node;
}

public DictionaryNode Search(DictionaryNode node, String word) {
    if (node == null || (node.word != null && node.word.equals(word))) {
        return node;
    }
    if (word.compareTo(node.word) < 0) {
        return Search(node.left, word);
    } else {
        return Search(node.right, word);
    }
}

public DictionaryNode FindMin(DictionaryNode node) {
    while (node.left != null) {
        node = node.left;
    }
    return node;
}
```

```
    public DictionaryNode Delete(DictionaryNode node, String word) {
        if (node == null) {
            return node;
        }
        if (word.compareTo(node.word) < 0) {
            node.left = Delete(node.left, word);
        } else if (word.compareTo(node.word) > 0) {
            node.right = Delete(node.right, word);
        } else {
            if (node.left == null) {
                DictionaryNode temp = node.right;
                node = null;
                return temp;
            } else if (node.right == null) {
                DictionaryNode temp = node.left;
                node = null;
                return temp;
            }

            DictionaryNode temp = FindMin(node.right);
            node.word = temp.word;
            node.meaning = temp.meaning;
            node.right = Delete(node.right, temp.word);
        }
        return node;
    }

    public void UpdateFile() {
    if (root != null) {
        try {
            BufferedWriter outFile = new BufferedWriter(new
FileWriter("C:\\Users\\dv\\Desktop\\Dictionary\\src\\dictionary\\Dic.txt"));
            WriteToFile(outFile, root);
            outFile.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error: Unable to open file Dictionary.txt for writing");
        }
    }
}
    public void WriteToFile(BufferedWriter outFile, DictionaryNode node) throws IOException {
        if (node != null) {
            WriteToFile(outFile, node.left);
            outFile.write(node.word + " " + node.meaning);
            outFile.newLine();
            WriteToFile(outFile, node.right);
        }
    }
```

```
public void InOrderPrint() {
      InOrderPrint(root);
   }

      public void InOrderPrint(DictionaryNode node) {
      if (node != null) {
         InOrderPrint(node.left);
         InOrderPrint(node.right);
      }
   }
```

5. INTERFACES:

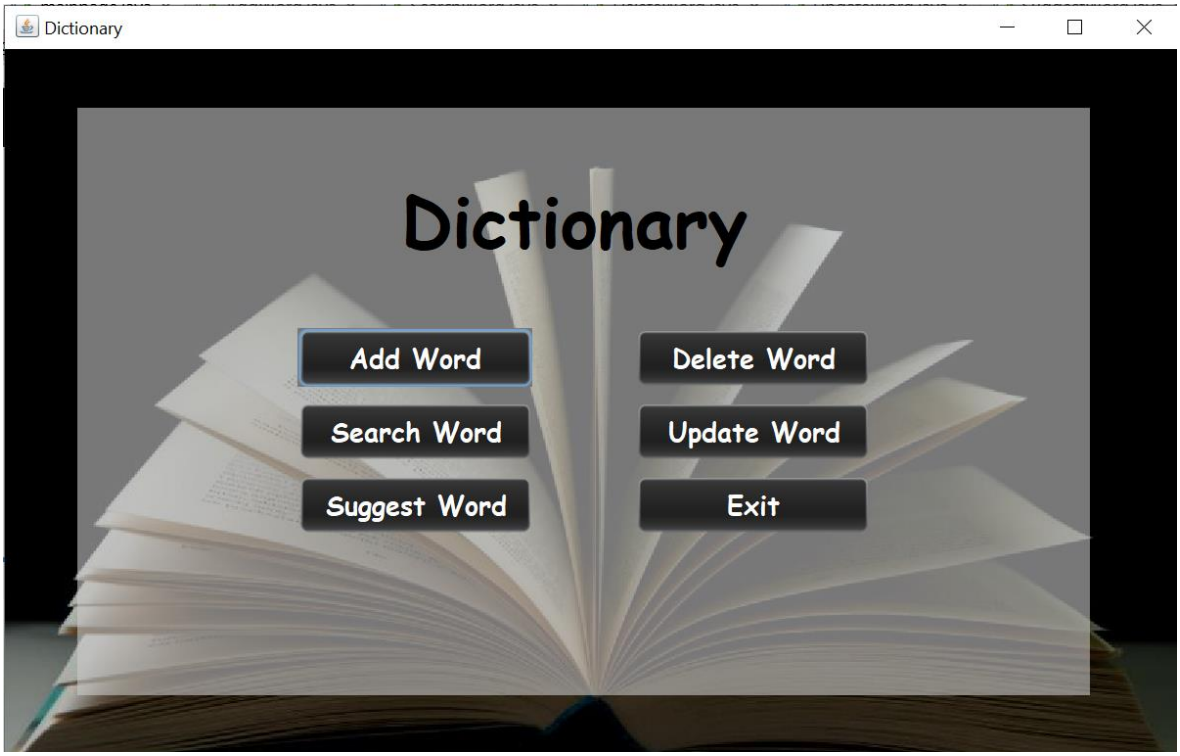

Figure 1 Main Page of Dictionary

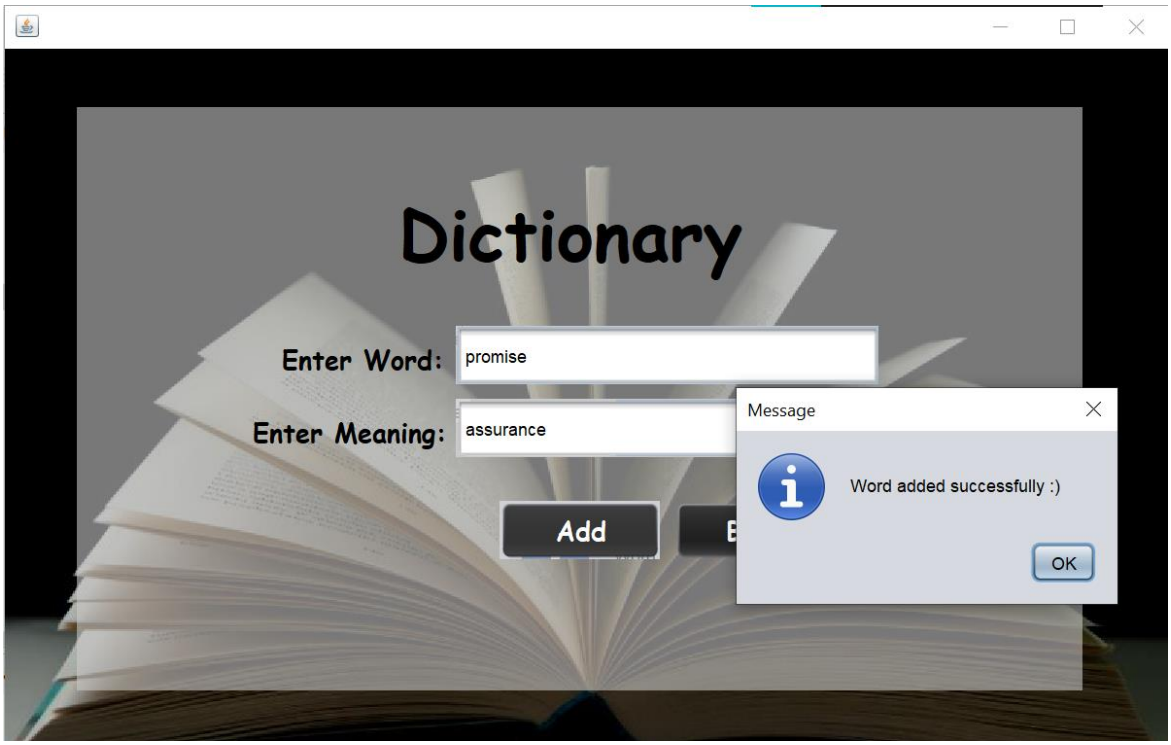

Figure 2 This interface adding new words in dictionary

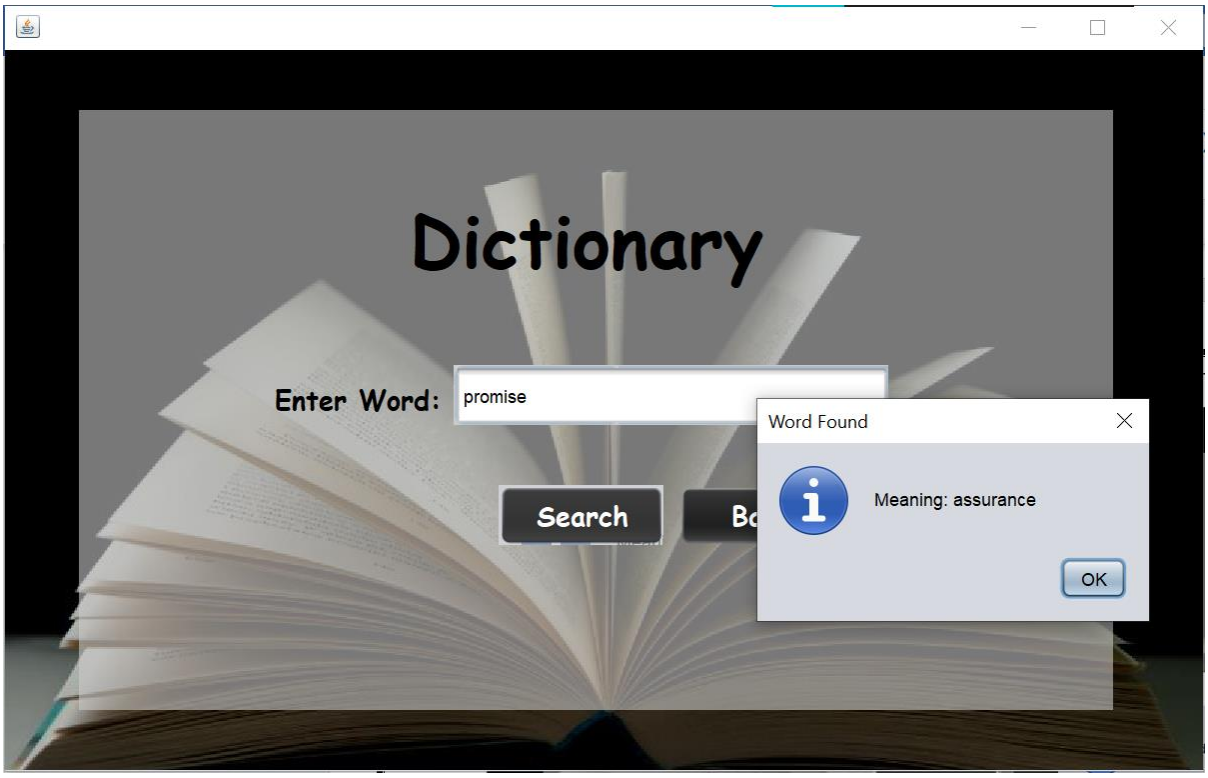

Figure 3 searching word in dictionary

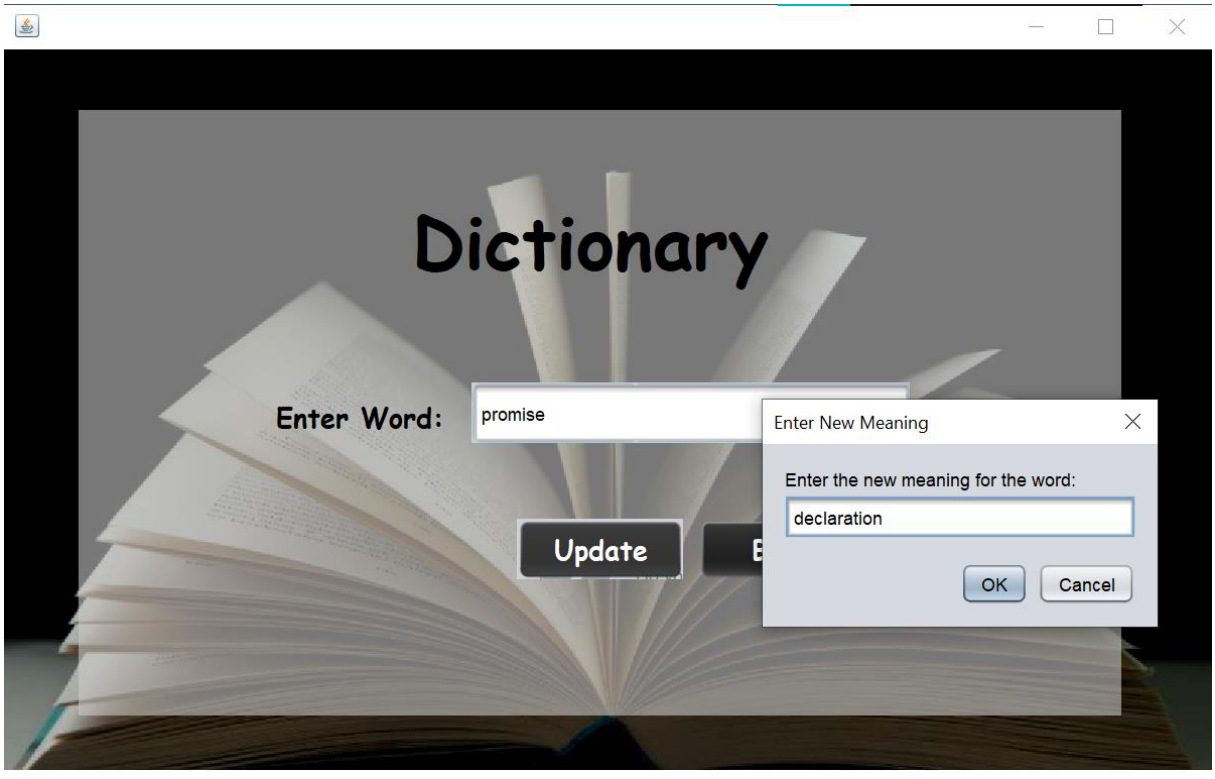

Figure 4 Updating word's meaning in dictionary

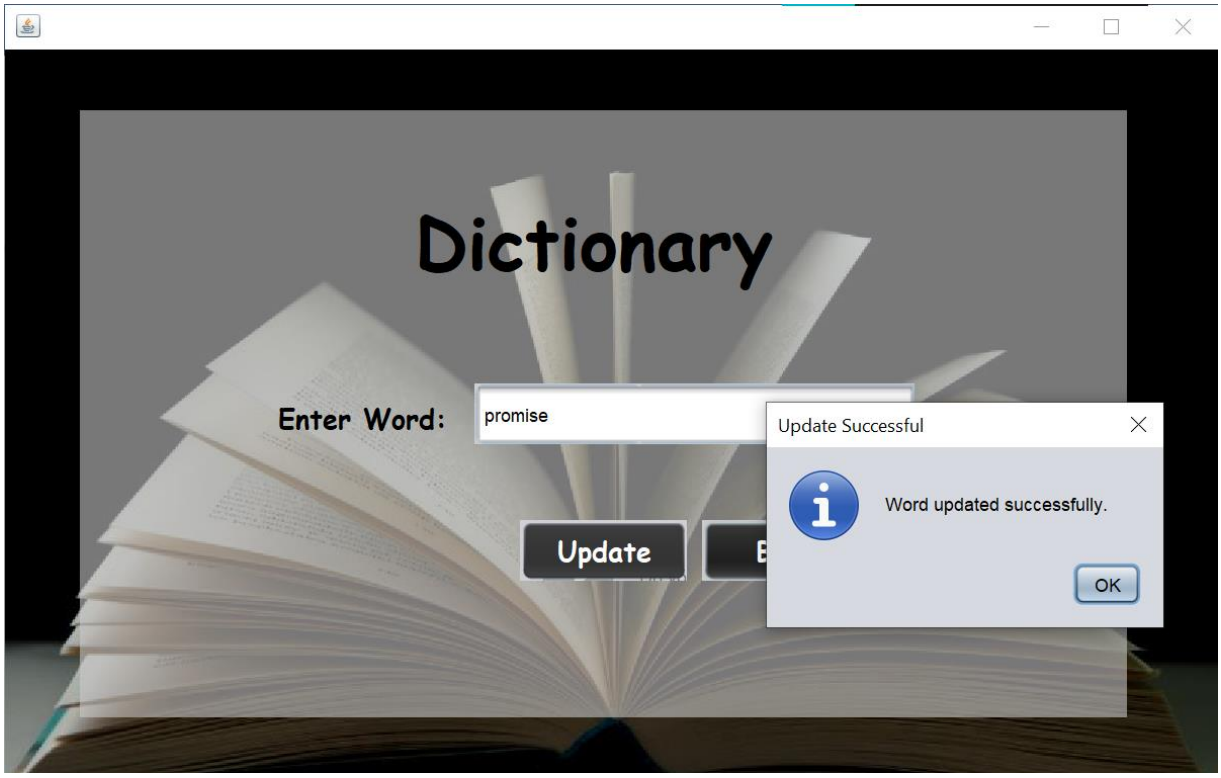

Figure 5 word's meaning updated

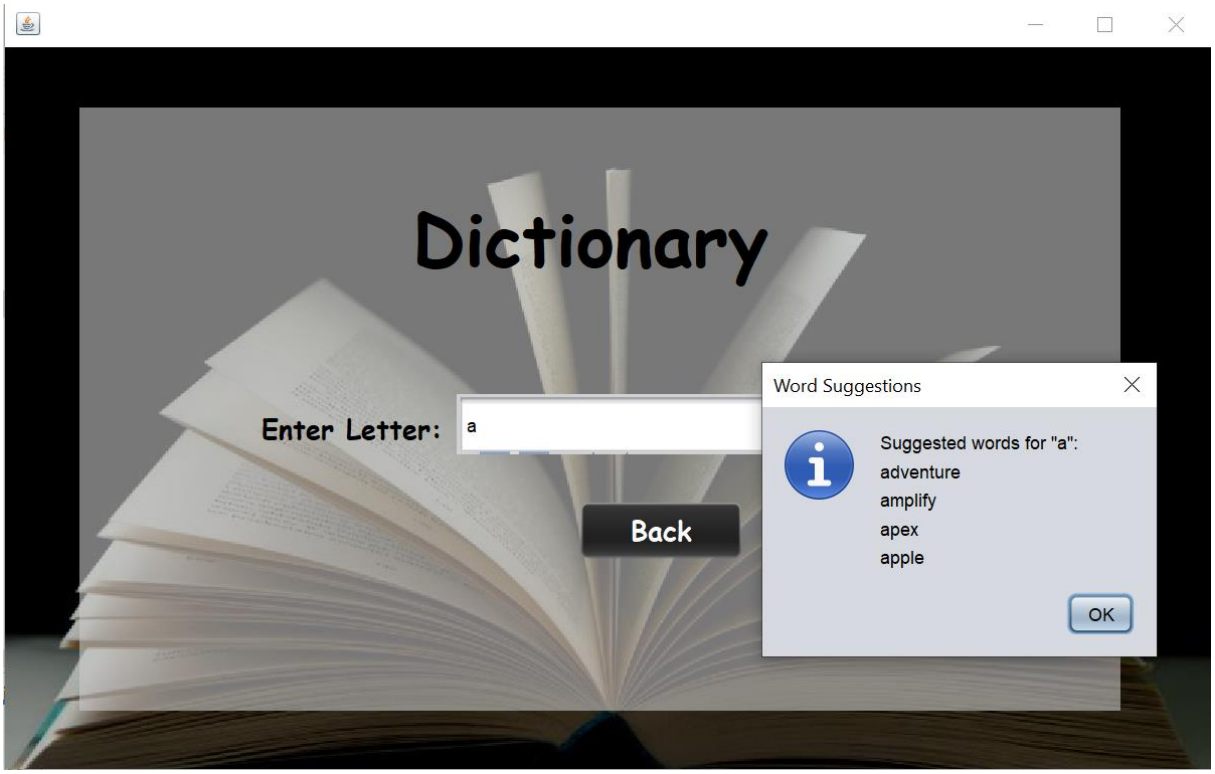

Figure 6 suggesting words from dictionary

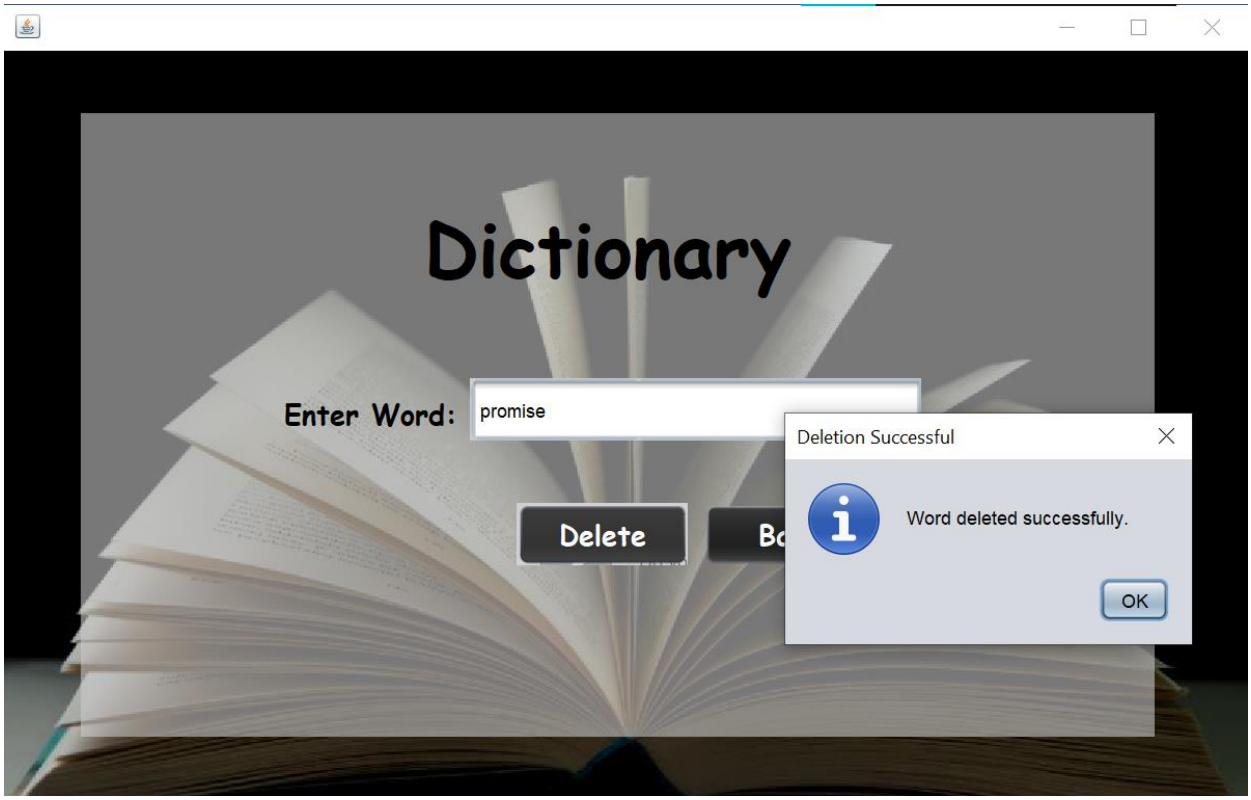

Figure 7 deleting word from dictionary

6. CONCLUSION:

In conclusion, the Dictionary project reimagines the traditional dictionary by introducing a dynamic, user-centric, and digital approach. Leveraging the power of Java and the efficiency of Binary Search Trees, this application provides an immersive experience for users to engage with language in a fluid and interactive manner. Whether exploring the depths of language or actively contributing to a personal lexicon, the Dictionary project offers a modern and efficient solution to language enthusiasts.