**DELHI TECHNOLOGICAL UNIVERSITY**

****

**COURSE: DISCRETE STRUCTURES (IT205)**

PROJECT REPORT

**Group members:**Aditi Singh(2K19/IT/005)

Anoushka Singh (2K19/IT/023)

ACKNOWLEDGEMENT

We, Aditi Singh(2K19/IT/005) and Anoushka Singh(2K19/IT/023) would like to express our heartfelt gratitude to our teacher Mrs. Swati Sharda Ma’am for giving us this wonderful opportunity to make a project on the topic “Spell checker using Tries ”. This project enabled us to learn so much about the applications of the simulation principles we learnt in class. It was only because of the kind guidance of our teacher that we were able to complete this project.

We would also like to thank our college, Delhi Technological University for giving us an opportunity to study an important subject like “Discrete Structures” which would help in our holistic development as an engineer.

Thanking You ,

Aditi Singh(2K19/IT/005)

Anoushka Singh(2K19/IT/023)

ABSTRACT

This project will help to check if spelling of word input by user is correct and to suggest correct spellings in case the input word is incorrect. This would be carried out using Tries data structure. We will also be displaying the time it takes to search for a particular word using tries and by Brute force method to show how fast it is to search for a word using tries.

A Trie is a special data structure used to store strings that can be visualized like a graph. It consists of nodes and edges like a tree. Each node consists of at max 26 children and edges connect each parent node to its children. These 26 pointers are nothing but pointers for each of the 26 alphabets in English language.

Tries are very useful as we can search a word in O(n) time where n is the length of a string/word we need to find.

Our code has been written in C++. GUI has been created using Qt.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1. | Abstract | 3 |
| 2. | Index | 4 |
| 3. | Objective | 5 |
| 4. | Implementation | 5 |
| 5. | Flowchart | 6 |
| 6. | Working of tries | 7-9 |
| 7. | Sample output | 10-11 |
| 8. | References | 12 |

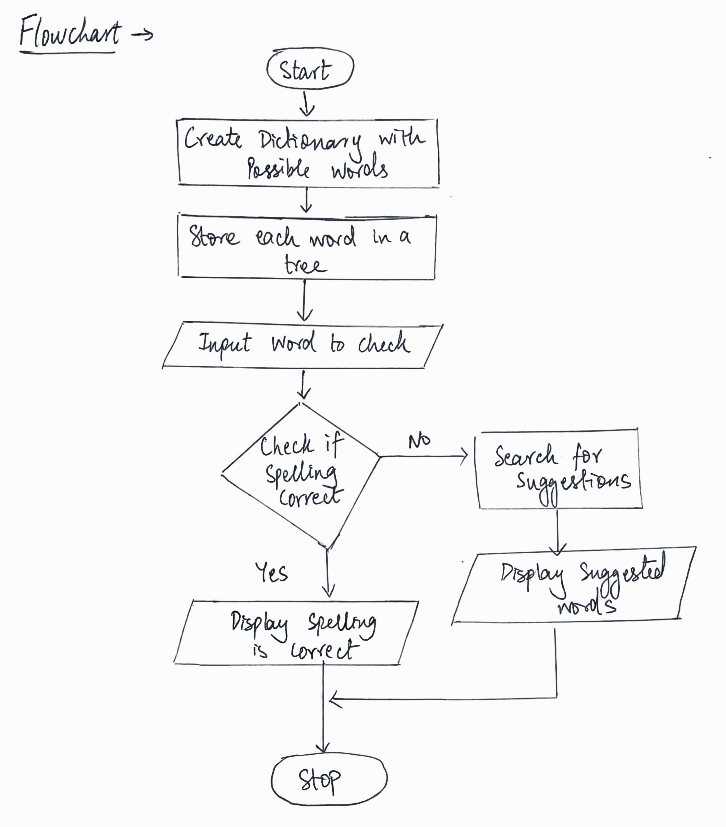
OBJECTIVE

* To check if spelling of word input by user is correct and to suggest correct spellings in case the input word is incorrect.
* To display the time it takes to search for a particular word using tries and by Brute force method to show how fast it is to search for a word using tries.

IMPLEMENTATION

* Create *wordlist.txt* file and add correct spelling words from the english dictionary . We added 70,000 words in this file.
* Create functions for-
* Inserting words in tries, searching for a word in tries
* Suggesting correct spelling suggestions if input word is incorrect.
* Calculating time taken to search using tries and using brute force method.
* Insert all words from *wordlist.txt* file to tries data structure.
* Take input of a word from user and call the search word function .
* Call suggest words function incase spelling of input word is wrong.

*(FLOWCHART ON NEXT PAGE)*



WORKING OF TRIES

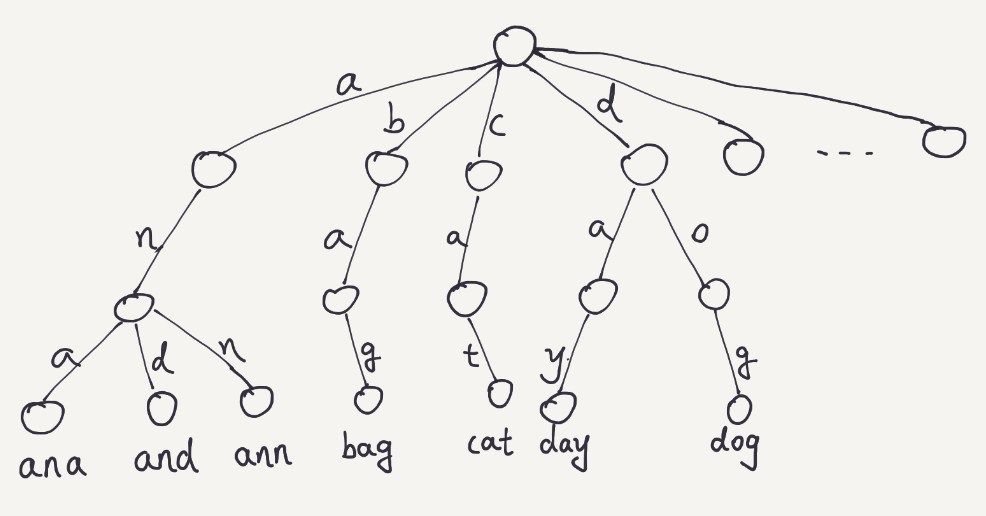
A trie is a tree-like data structure whose nodes store the letters of an alphabet. By structuring the nodes in a particular way, words and strings can be retrieved from the structure by traversing down a branch path of the tree.

Each trie has an empty root node, with links to other nodes — one for each possible alphabetic value.*(In our code each node has 256 children -one for every ASCII character)*

Each node has the following parameters-

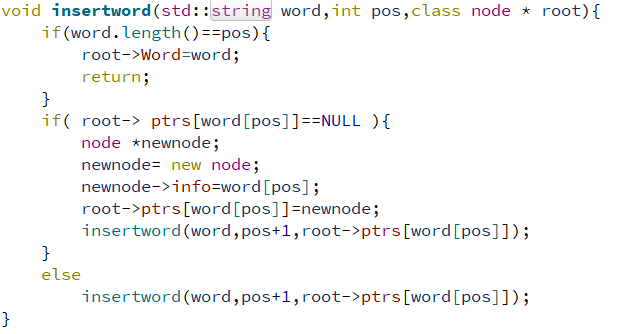
1. **char info-** This stores a character . info is initially set to NULL on the creation of the node.
2. **string word-** While inserting words in tries , a meaningful word is formed once the node containing the last character of that word is inserted . This node stores that particular word in its *string word* parameter. Initially *string word* parameter is set to “” on creation of a node.
3. **node\* ptrs[256]-** This is an array of pointers to children nodes . Initially all pointers are set to NULL.

**HOW WORDS ARE STORED IN TRIES-**

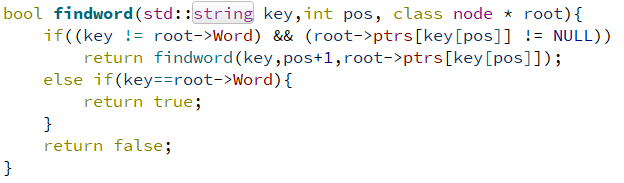


**Functions-**

**Insertword() function**- initially checks to see if the word is present in the trie or not, if it is not present how many characters continuously match from the root of the node to the word that we are inserting. After that we insert the remaining characters into the trie.

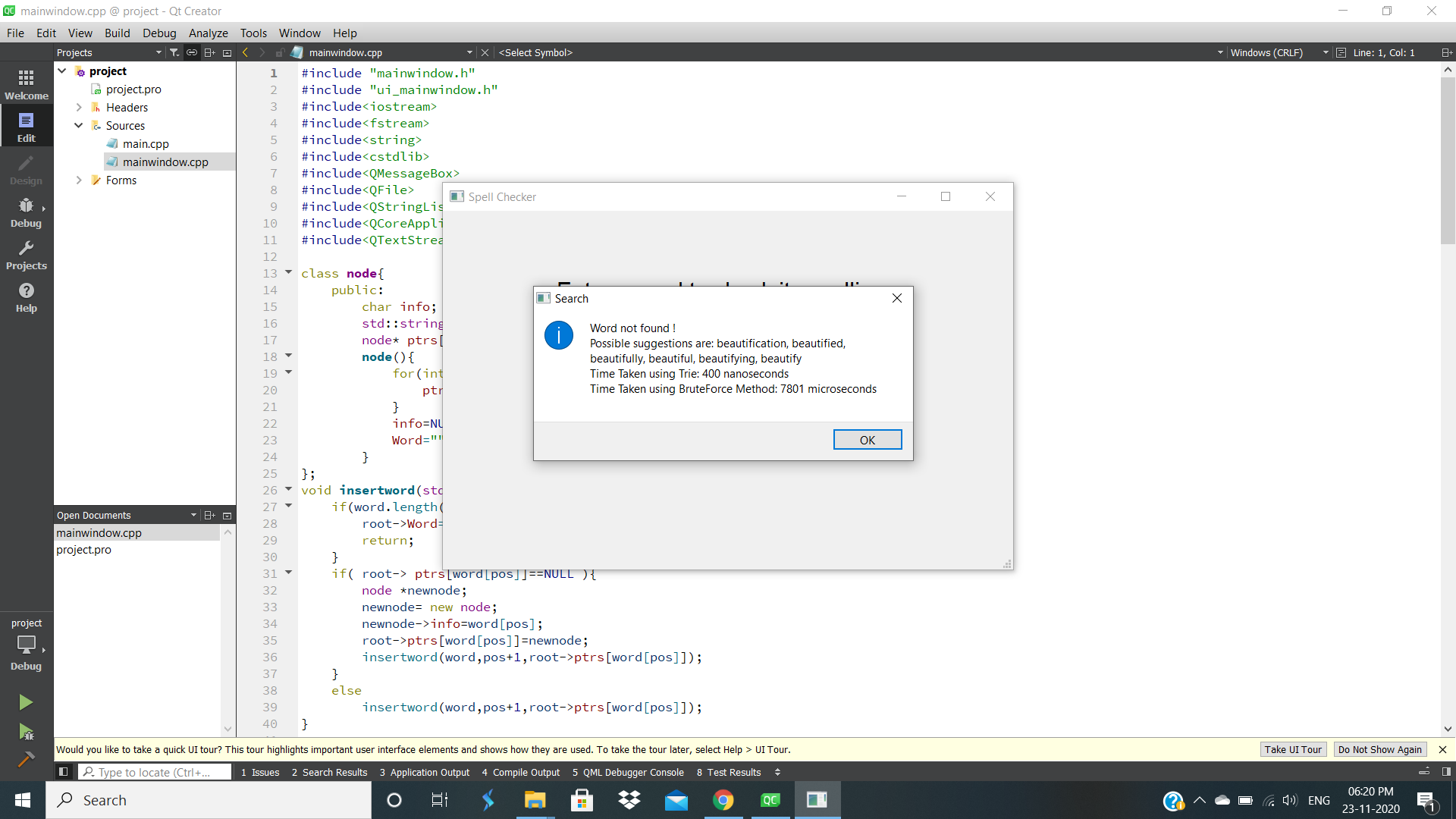
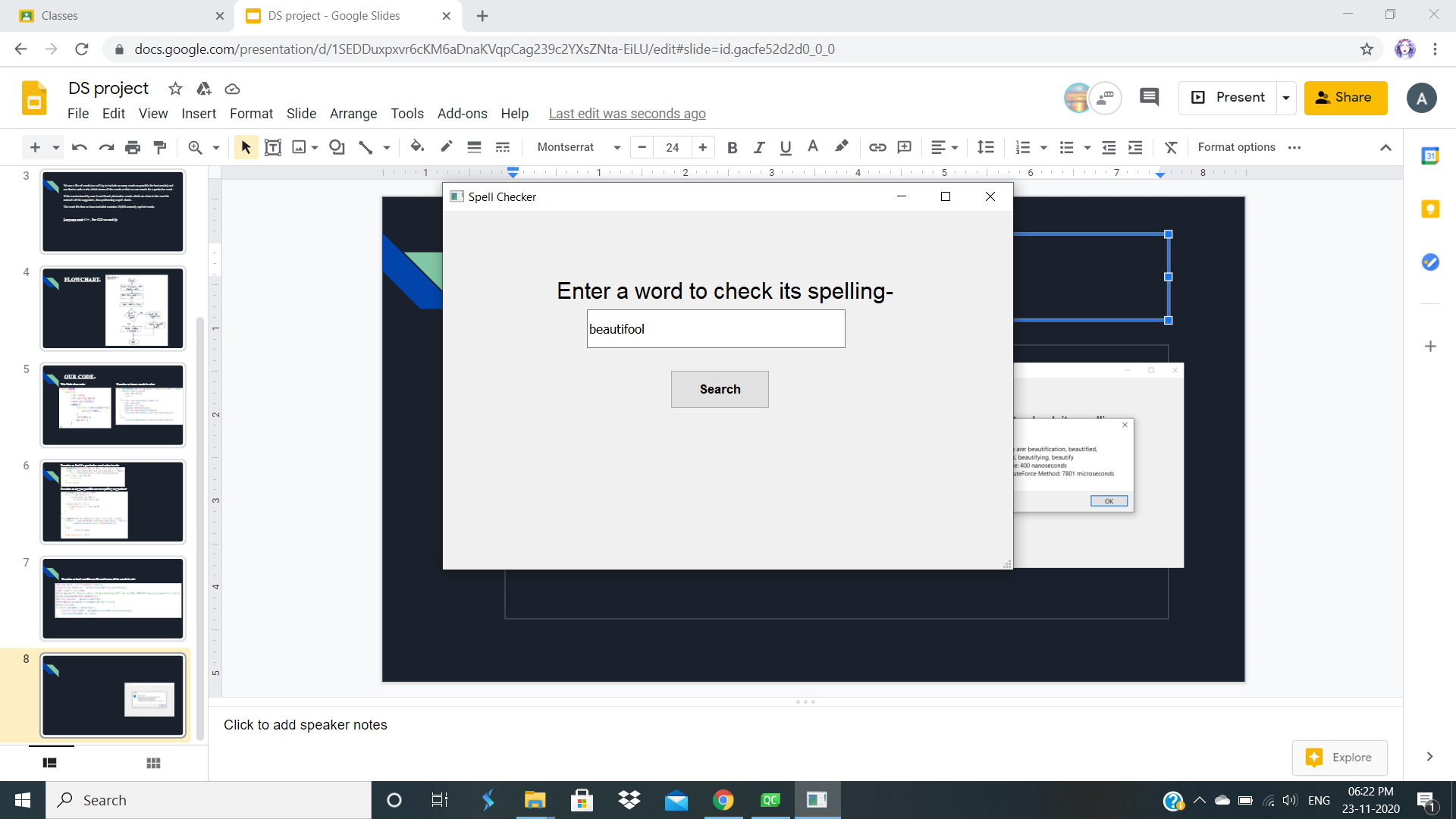


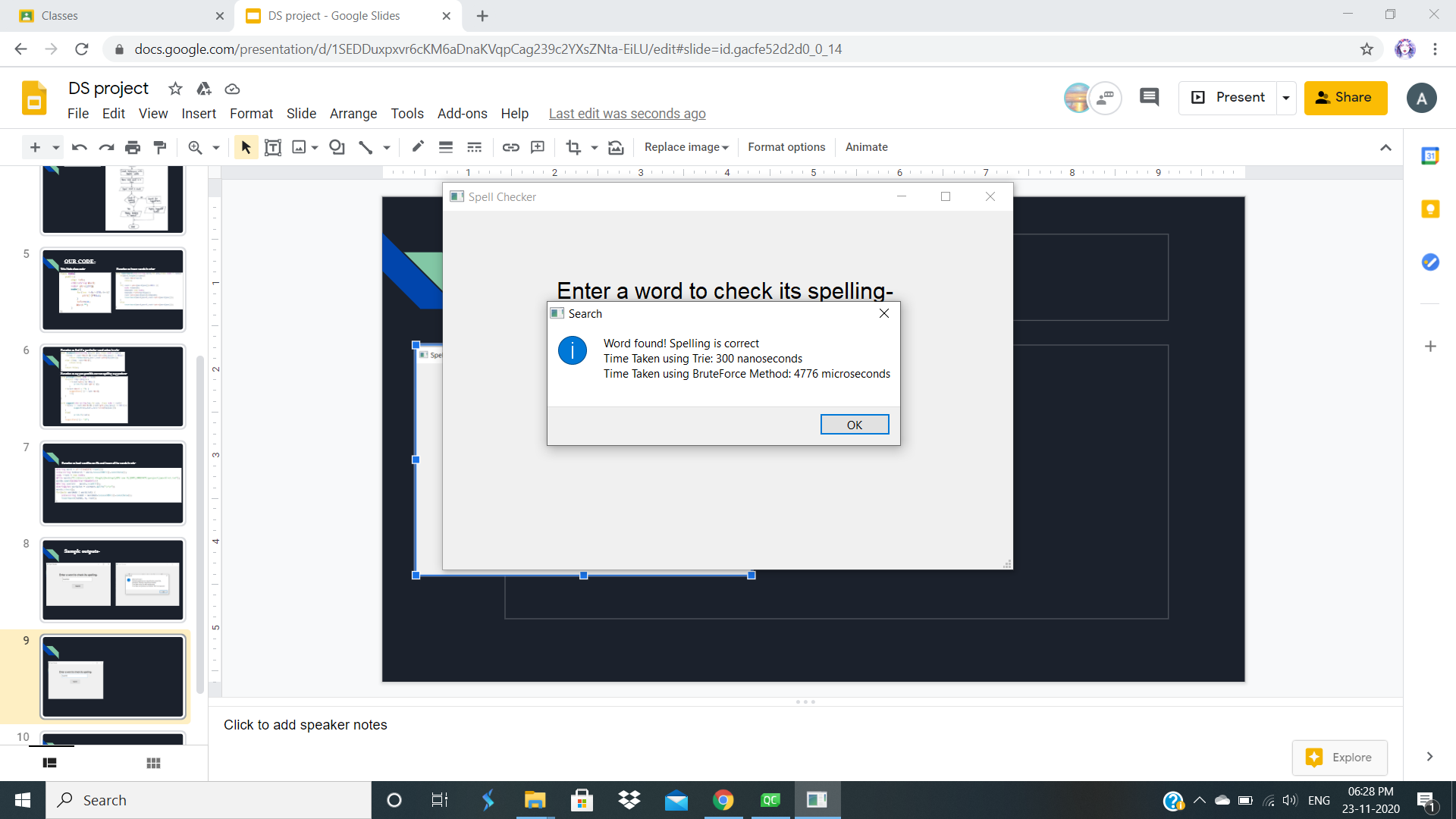
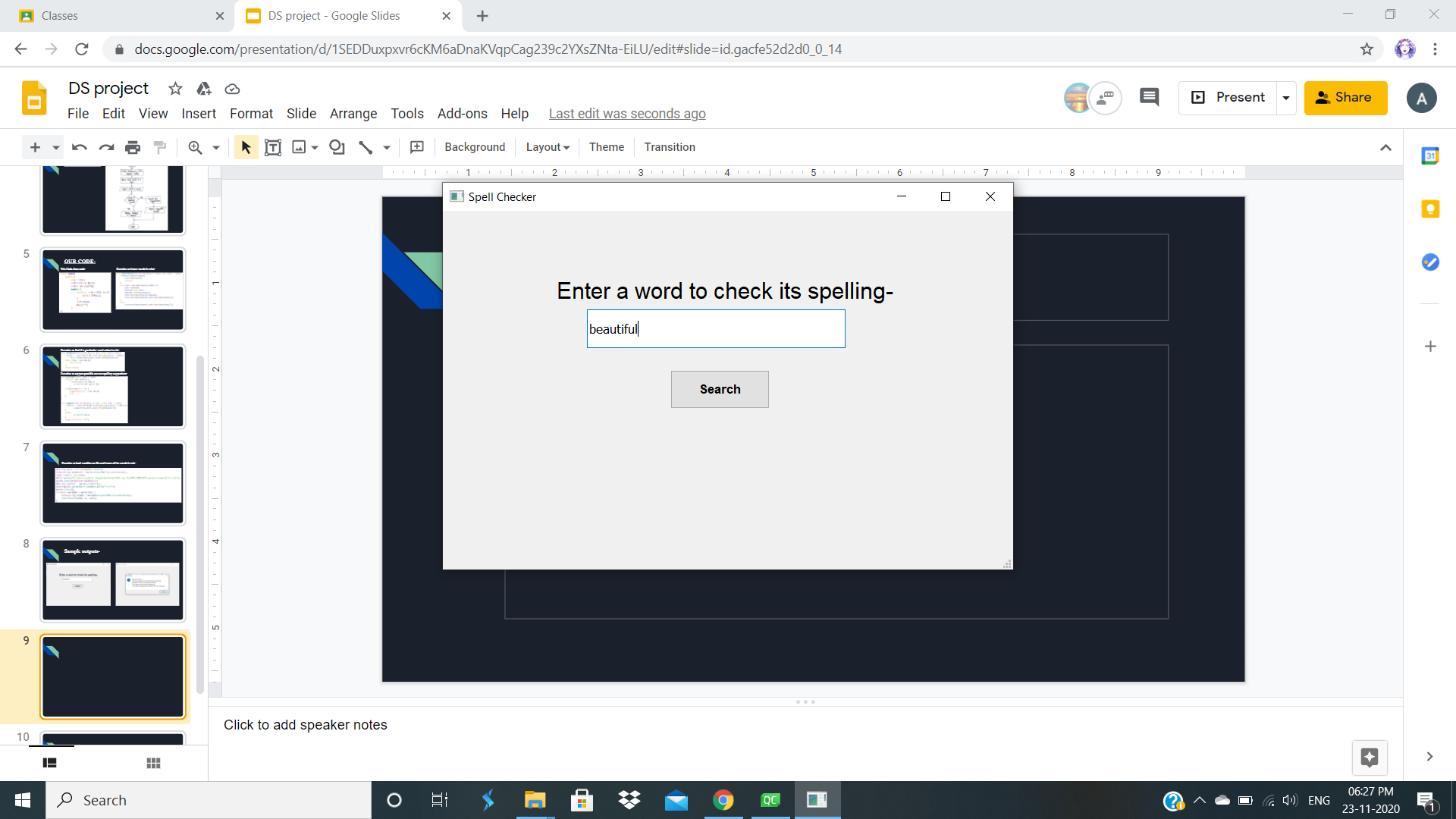
**Findword() function**- starts matching from the first character of the node and goes on, if there is a mismatch, it terminates and returns false, meaning the word is not present in the trie.



**Suggest() function-** is used to suggest alternative words present in the dictionary if the word entered by the user is not found.

SAMPLE OUTPUTS-





REFERENCES

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.736.3672&rep=rep1&type=pdf>

<https://medium.com/basecs/trying-to-understand-tries-3ec6bede0014>

Github Link- <https://github.com/aditijaipur123/spell-checker-project>