# Digital Destroyers

## Spell checker

## PROJECT ID:P8

## SECTION-A


Team members:

- SAMARTH AGRAWAL:202301040
- MANU PARDHAN:202301020
- DEV MAKWANA:202301149
- MANASVI RATHORE:202301108


Github link:

https://github.com/DIGITALDESTROYERS/spellcheck.git

### Time complexity:

- Insertion (insertword):O(n),where n  is the length of the word being inserted. The algorithm iterates through each character of the word and performs constant-time operations for each character.

- Search (searchword):O(n),where n is the length of the word being searched. Similar to insertion, it iterates through each character of the word.

- Deletion (deleteword): O(n),where n is the length of the word being deleted. It also traverses through the characters of the word.

- Update (updateword):O(n),where n The update operation is essentially a combination of deletion and insertion, so its time complexity is O(n),n is the length of the word being updated.

## SPACE COMPLEXITY:

- Trie Data Structure:O(number of nodes),where the number of nodes depends on the number of characters and the structure of the words inserted into the Trie.

- Total Space: Since each node in the Trie consumes a constant amount of memory, the space complexity depends on the number of nodes created during the insertion of words. In the worst case, if there are N words with a maximum of M characters each, the space complexity is O(N . M).

# pseudocode:

```
Class Trie:
  TrieNode root
  Constructor Trie():
    Initialize root node
  Method insertword(word):
    Initialize currNode as root
    For each character c in word:
        If currNode's child at index (c - 'a') is NULL:
         Create a new TrieNode and set it as the child at index (c - 'a')
         Move currNode to its child at index (c - 'a')
    Set currNode's iscomplete to true
  Method searchword(word):
    Initialize currNode as root
    For each character c in word:
      If currNode's child at index (c - 'a') is NULL:
        Return false
```

```
        Move currNode to its child at index (c - 'a')
    Return currNode's iscomplete
Method deleteword(word):
      Initialize currNode as root
    For each character c in word:
        If currNode's child at index (c - 'a') is NULL:
            Return false
        Move currNode to its child at index (c - 'a')
      set currNode's iscomplete to false
    Return true
Method updateword(oldword, newword):
    Call deleteword(oldword)
    Call insertword(newword)
```