**Adavance Data Structures Lab (20XD37)**

**Semester – III Final Package**

**Abstract**

**---------------------------------------------------------------------------**

**Team Members:**

22PD05 – Akshaya S

22PD16 - Kaviya NS

22PD35 – Sujan S

**Topic:**

SEARCH ENGINE (auto complete)

**Data Structures Used:**

TRIES

**Language Used**:

Python

**Libraries Used:**

time – The time module provides various time-related functions, including methods for working with time values, measuring time intervals, and handling time-related operations.

**Classes and Functions Used:**

1. **char\_to\_index(c)**: This function takes a character `c` as input and maps it to an index within the alphabet. Lowercase letters map to indices 0-25, and numbers map to indices 26-35.

2. **TrieNode**: This class defines a node in the trie data structure. Each node has an array of children nodes and a boolean flag `isWordEnd` indicating whether the node represents the end of a word.

3. **get\_node()**: This function creates and returns a new trie node.

4. **User**: This class represents a user and stores their search history. It has a list called `history` to store search history.

5. **insert\_user\_history(user, query, suggestions)**: This function takes a user, a query, and a list of suggestions. It appends the query and suggestions to the user's search history.

6**. insert(root, key)**: This function inserts a key (word) into the trie. It converts the key to lowercase and iterates through its characters, creating nodes as needed in the trie.

7. **is\_last\_node(root)**: This function checks if a given node is the last node (has no children).

8. **suggestions\_rec(root, curr\_prefix, limit, results)**: This is a recursive function used to find auto-suggestions. It traverses the trie and adds suggestions to the `results` list.

9. **print\_auto\_suggestions(root, query, limit=None)**: This function starts the process of finding and printing auto-suggestions. It converts the query to lowercase, searches for the query in the trie, and then calls `suggestions\_rec` to find suggestions based on the query prefix. It also removes duplicate suggestions and prints the results.

10. **main()**: This is the main function of the program. It performs the following steps:

- Asks the user whether case sensitivity should be enabled.

- Initializes the trie (`root`) and a user object (`user`).

- Reads a list of words from a file ("words.txt") and inserts them into the trie.

- Prompts the user to enter a search query and the maximum number of suggestions to display.

- Measures the time taken for the search.

- Calls `print\_auto\_suggestions` to find and print suggestions.

- Inserts the query and the number of suggestions into the user's search history.

- Finally, it prints the user's search history.

11. **if \_\_name\_\_ == "\_\_main\_\_"**: is a standard Python construct that ensures the `main()` function is executed only if this script is run as the main program.

**ABSTRACT :**

The provided Python code implements a Trie-based Auto-Suggestion System. This system enables users to input a query, and it offers word or phrase suggestions based on the input query's prefix. The key components of this system include a Trie data structure for efficient word storage and retrieval and a user history feature that keeps track of previous search queries.

The code is structured as follows:

1**. Character Indexing**: The code handles both lowercase letters (a-z) and numbers (0-9) by mapping them to indices within an alphabet of size 36. This flexibility allows users to search for words or phrases that may contain letters and numbers.

2. **Trie Data Structure**: The `TrieNode` class represents nodes in the Trie data structure. Each node has children nodes, creating a tree-like structure. The `isWordEnd` flag indicates the end of a word within the Trie.

3. **User History**: The `User` class allows users to perform searches. It maintains a history of search queries and the corresponding suggestions. Users can also choose whether searches should be case-sensitive or not.

4. **Search and Suggestion**: The heart of the system lies in the `print\_auto\_suggestions` function. It converts the user's query to lowercase and uses the Trie to find and display suggestions based on the query's prefix. The number of suggestions displayed is limited by the user's choice.

5**. Main Function**: The `main` function orchestrates the entire system. It provides a command-line interface for users to specify their case-sensitivity preference, performs the Trie initialization with a predefined word list, takes user input for a search query, and then returns suggestions based on the query. The system also records the time taken for the search and appends the query and the number of suggestions to the user's search history.

This code is a practical implementation of an auto-suggestion system, which can be used in various applications, such as text editors or search engines. It showcases the efficiency of Trie data structures for word retrieval and provides users with a history of their search queries. The system can be extended and customized to suit specific requirements and preferences.