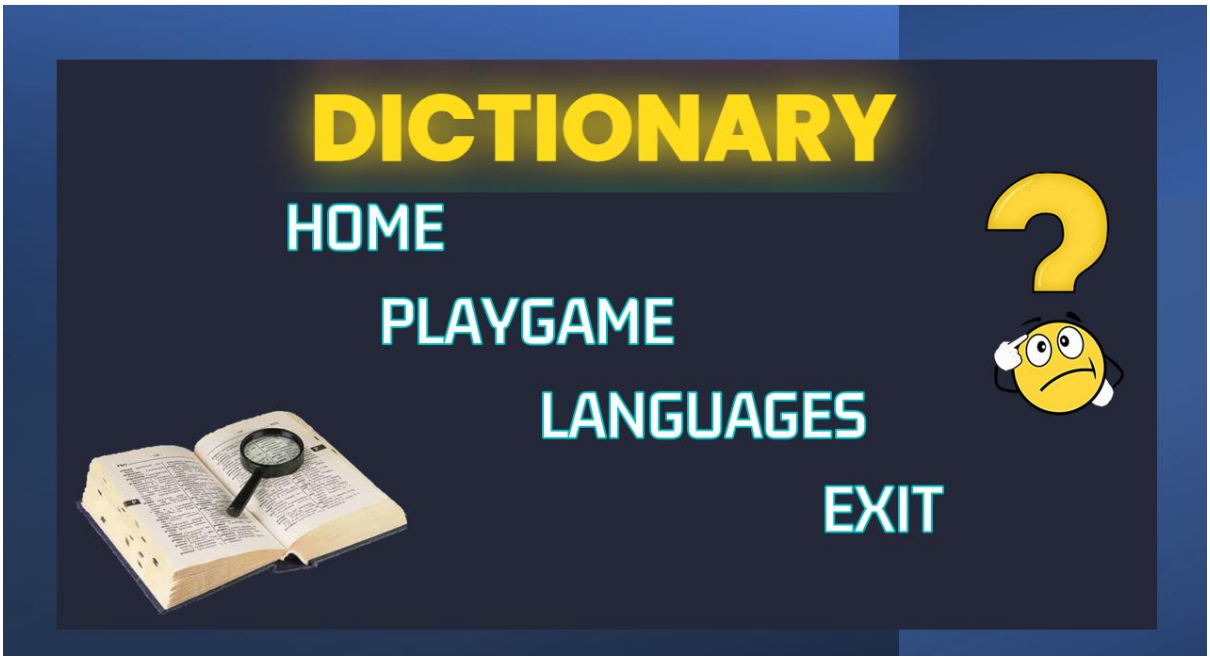
CS163 - Data Structures
Final Project
– Dictionary –

- **Team ID:** 10
- **Class:** 21APCS1
- **Member:**
    + *Nguyễn Đức Hưng: 21125042*
    + *Nguyễn Trọng Nghĩa: 21125154*
    + *Trần Thiên Phúc: 21125090*
    + *Diệp Tường Nghiêm: 21125155*

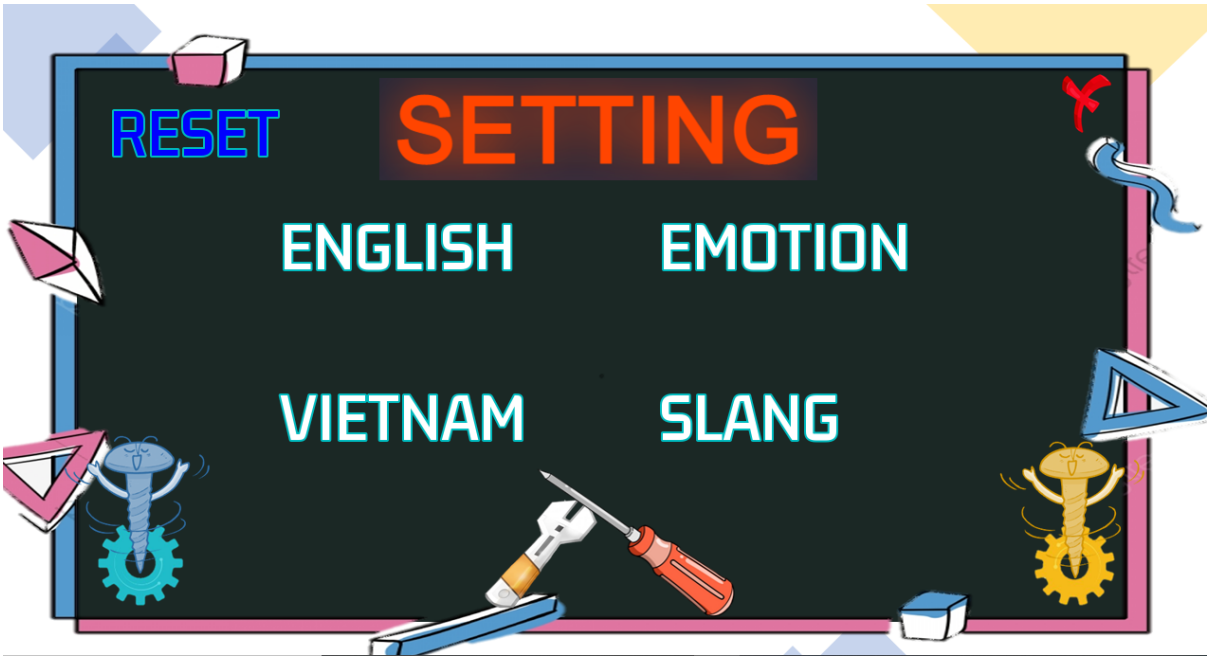
I. DESIGN AND FUNCTION

1. Main Menu:



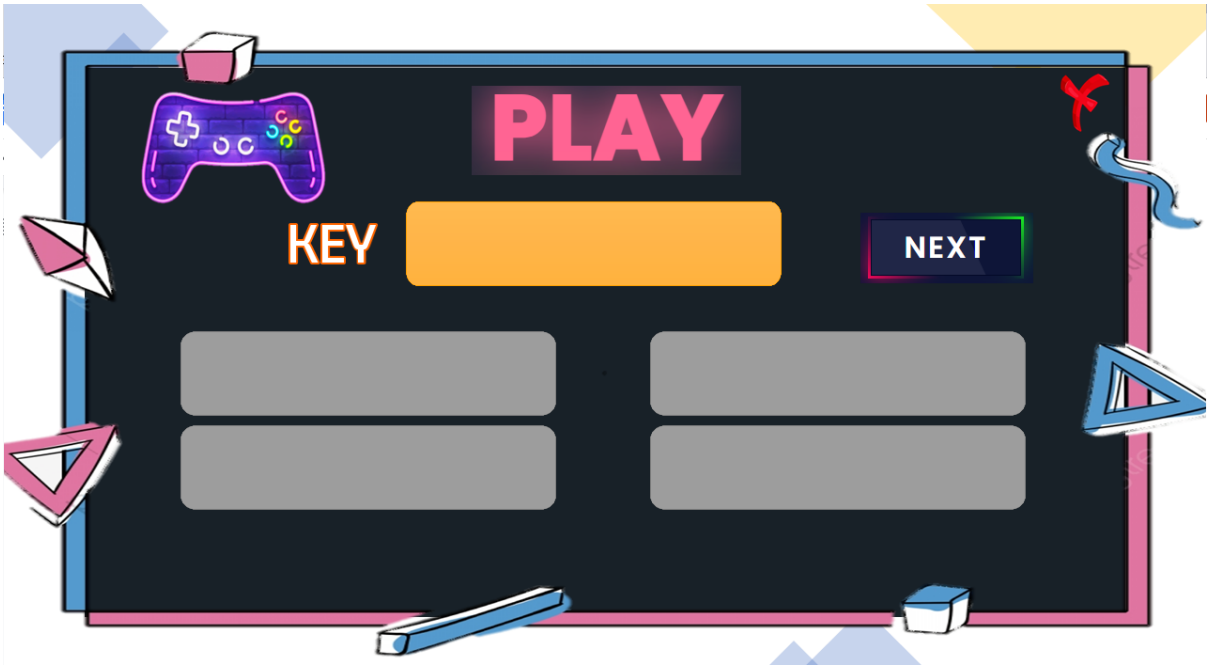
- **HOME**: switch datasets and reset datasets
- **PLAYGAME**: display one key and four definitions or one definition and four keys to choose
- **LANGUAGE**: lookup meaning, see history and favorite words
- **EXIT**: close the program

2. Select Dataset:

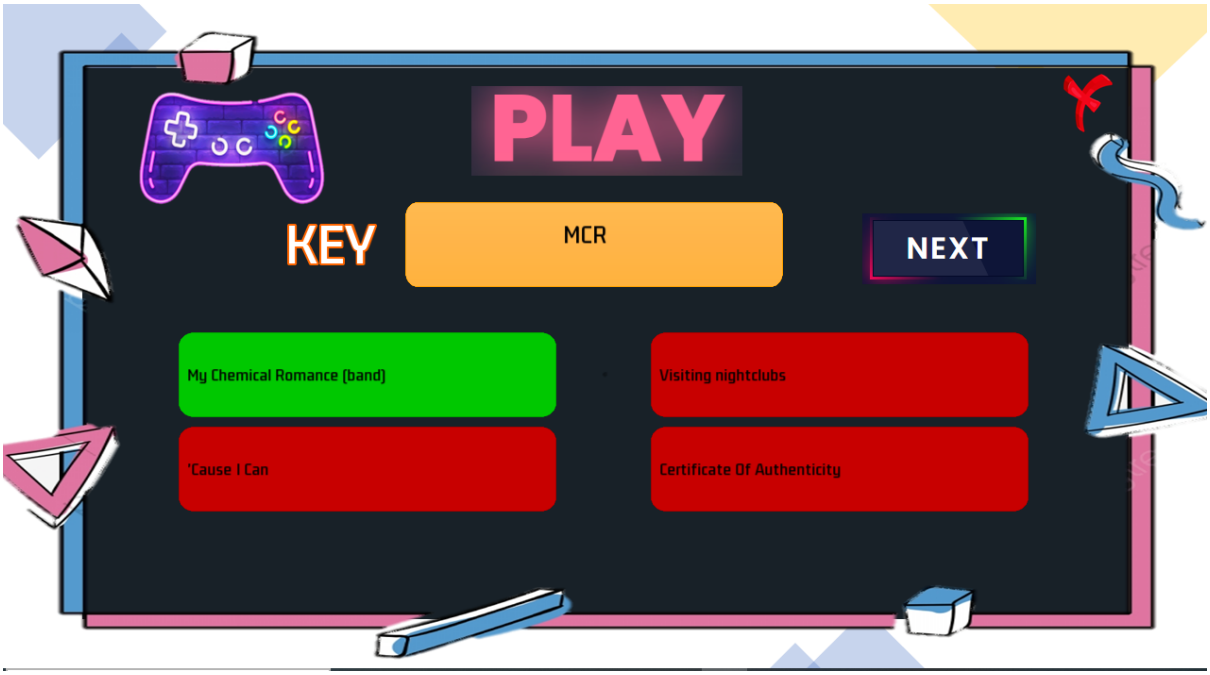


- **RESET**: return the original datasets choosing
- **ENGLISH**, **EMOTION**, **VIETNAM**, **SLANG**: four datasets (left-kick to choose)

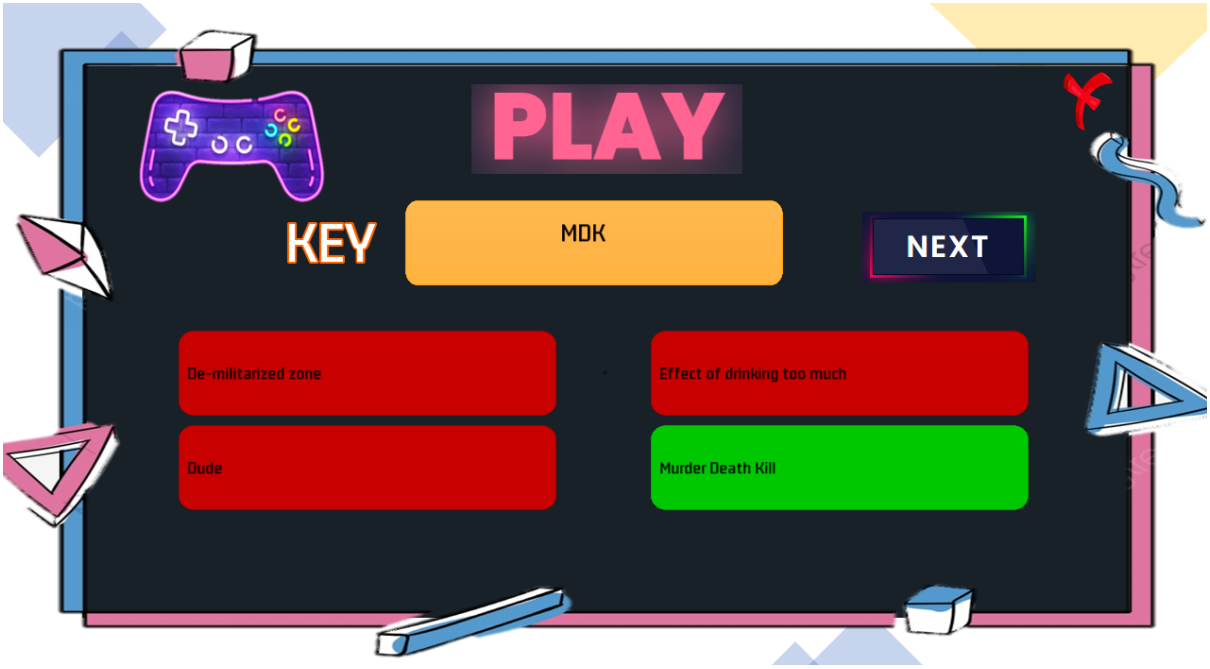
3. Game:



- **NEXT**: click here to generate words to practice
- **KEY**: return a random word when we press the next button
- **4 EMPTY GRAY BOXES**: return 1 correct meaning of key and randomize 3 trash meanings.

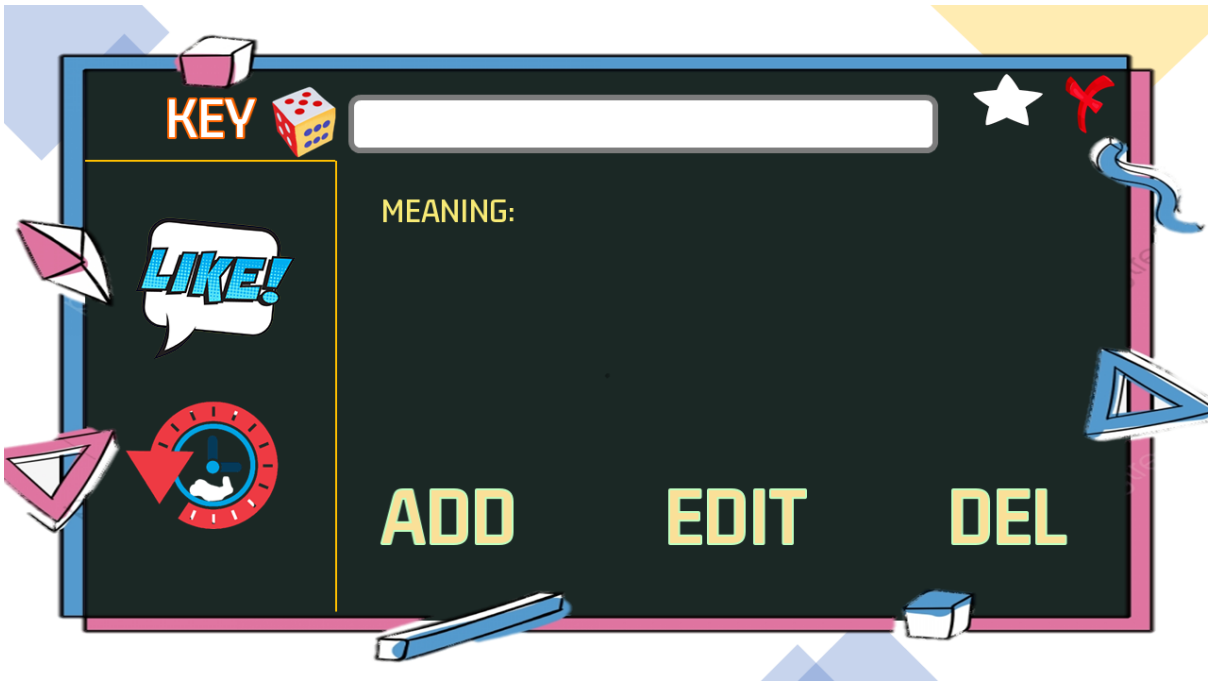


Example 1: With Key “MCR” the correct answer “My Chemical Romance [band]” will turn green when we click on it. Conversely, the remaining answers will turn red.

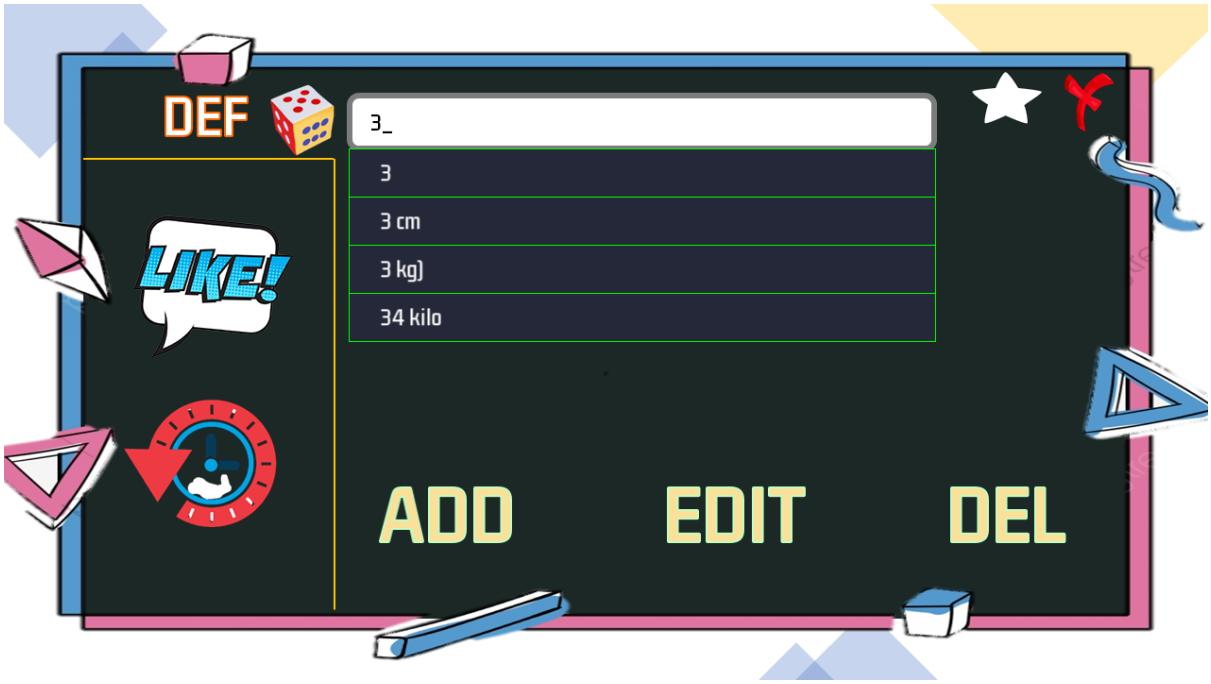


Example 2: With Key “MDK” the correct answer “Murder Death Kill” will turn green when we click on it. Conversely, the remaining answers will turn red.

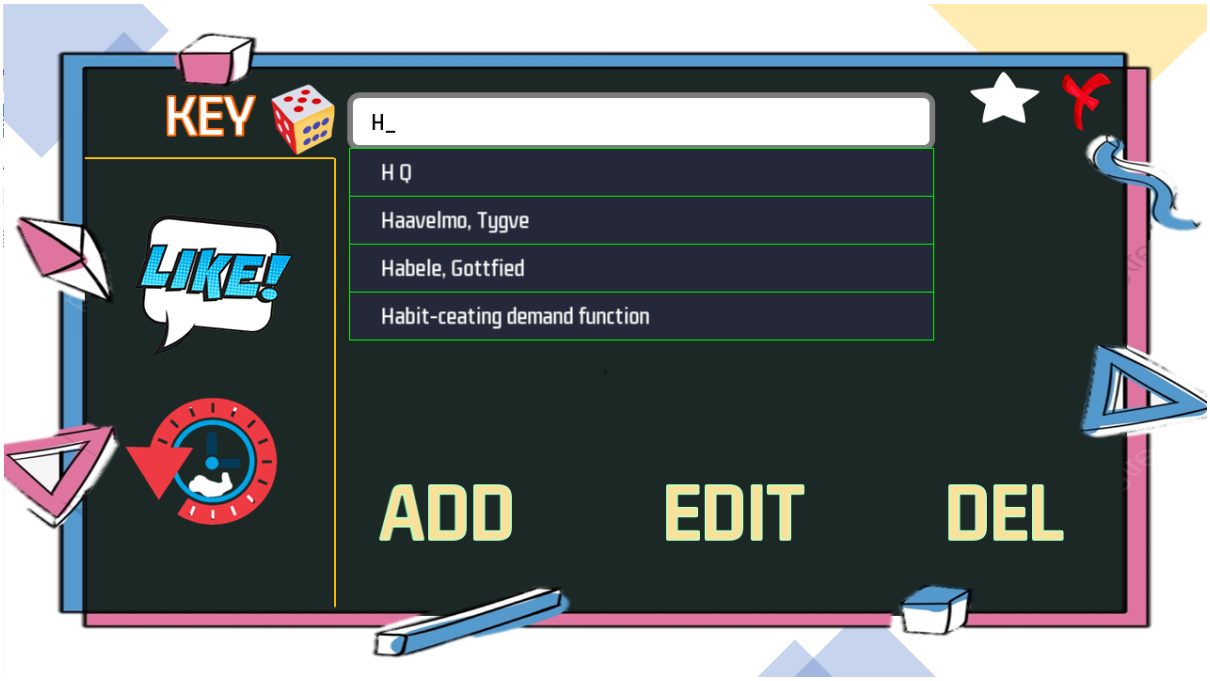
4. Words Look-Up Page



Interface of **LANGUAGE**
KEY: change from KEY to DEFINITION or change from DEFINITION to KEY
STAR: add word to favorite list
LIKE: see the favorite list
HISTORY: see the history list
DICE: lookup random word
ADD: add new word or add new meaning of word
EDIT: change the meaning of word
DEL: delete word or meaning of word

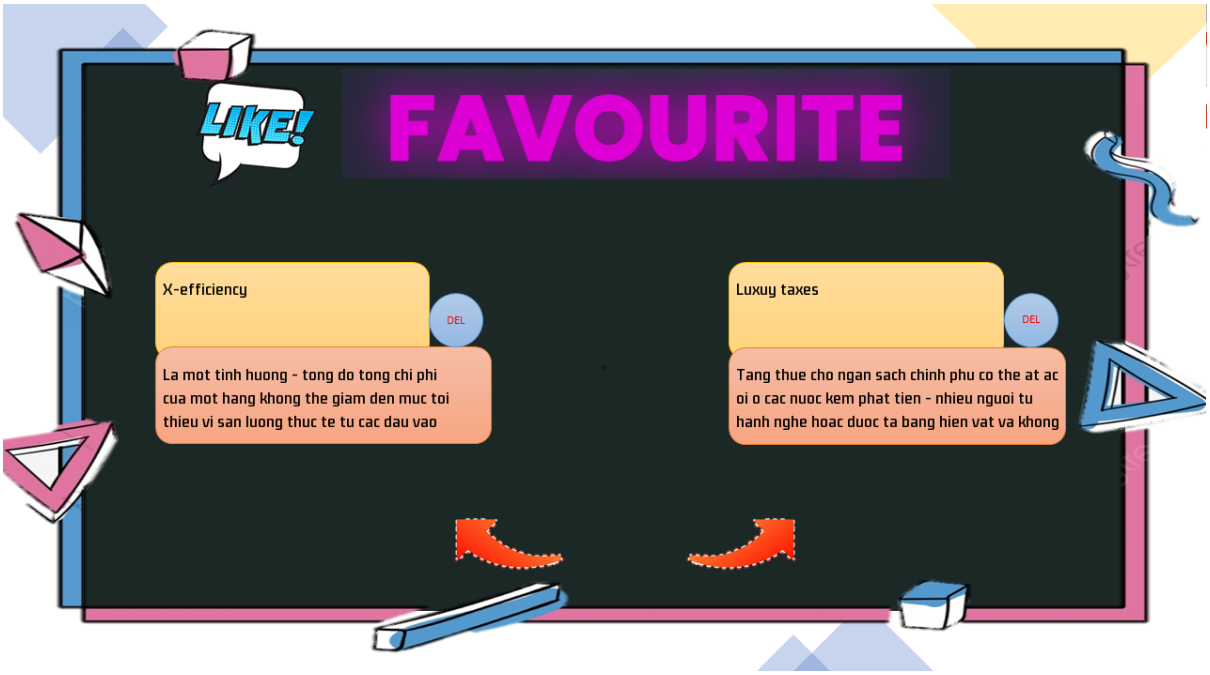


(Lookup from definition to key)

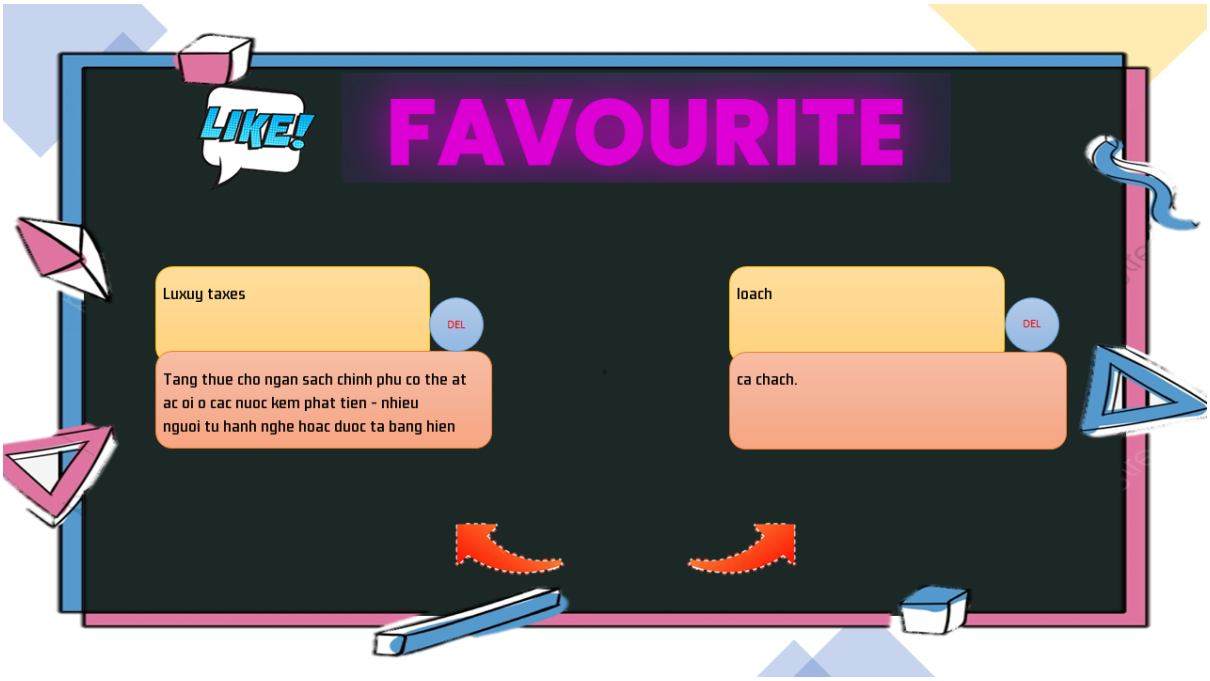


- Lookup from **key** to **definition**
- Click dice to lookup random word

5. Adding Word To Favorite

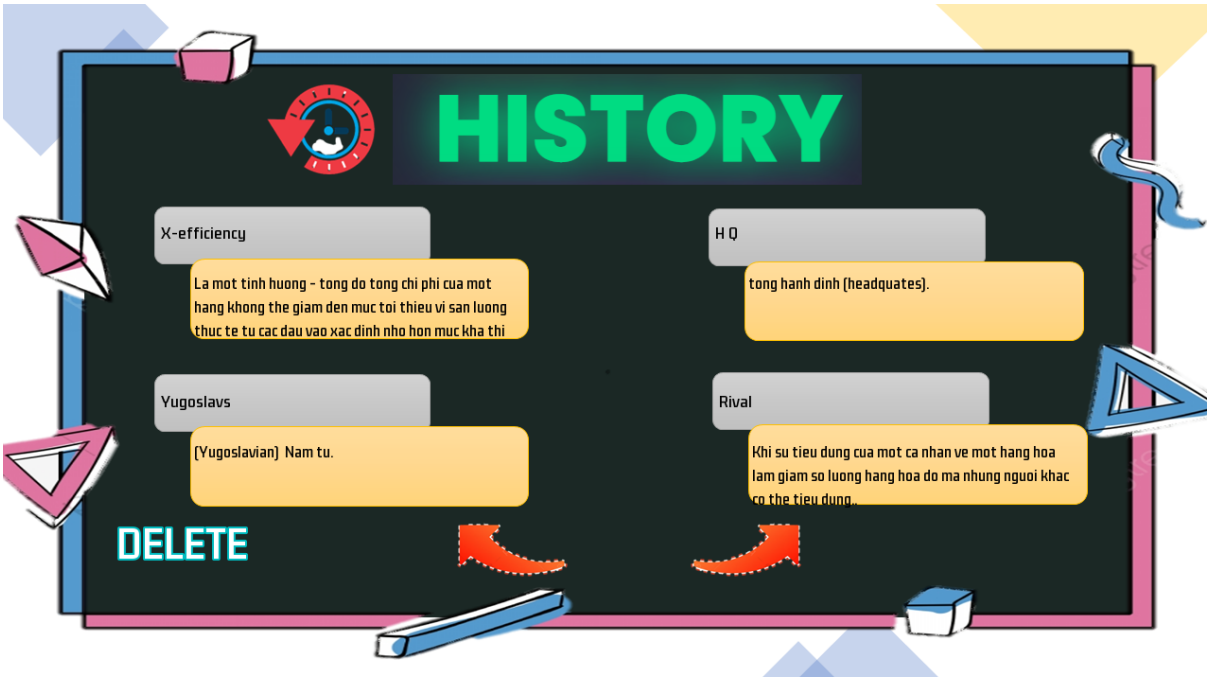


Click **STAR** to add into favorite word or **DEL** (blue button) to delete favorite word



(Result after delete favorite word)

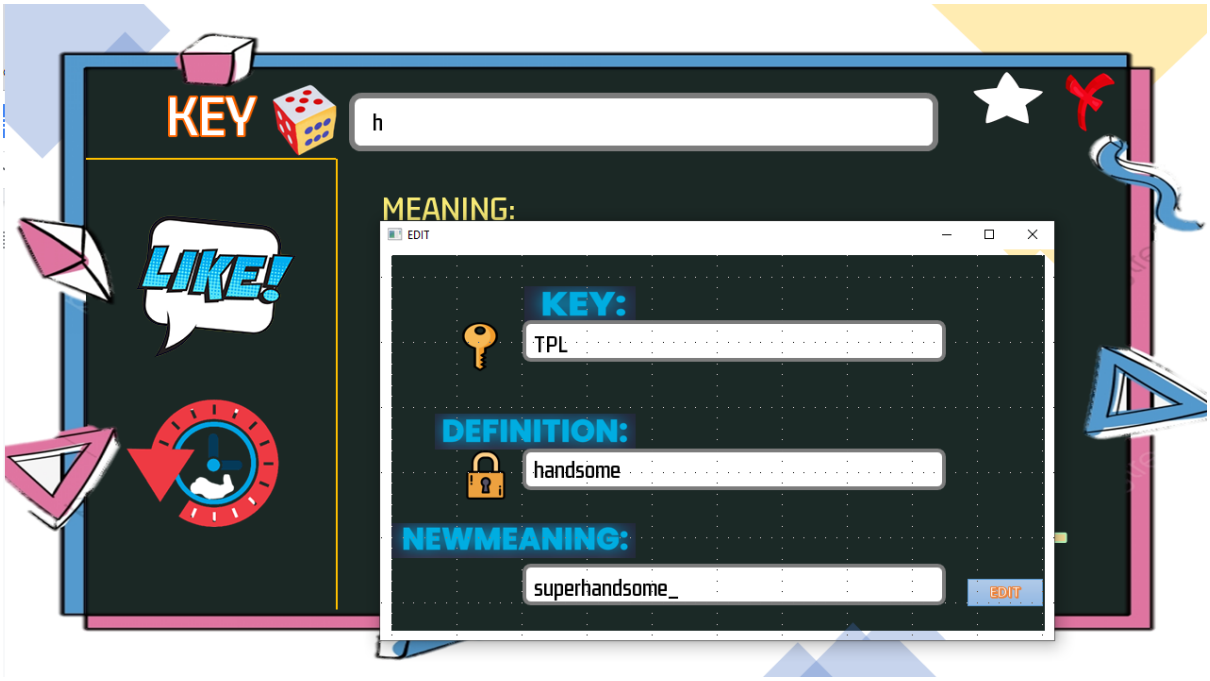
6. History Page



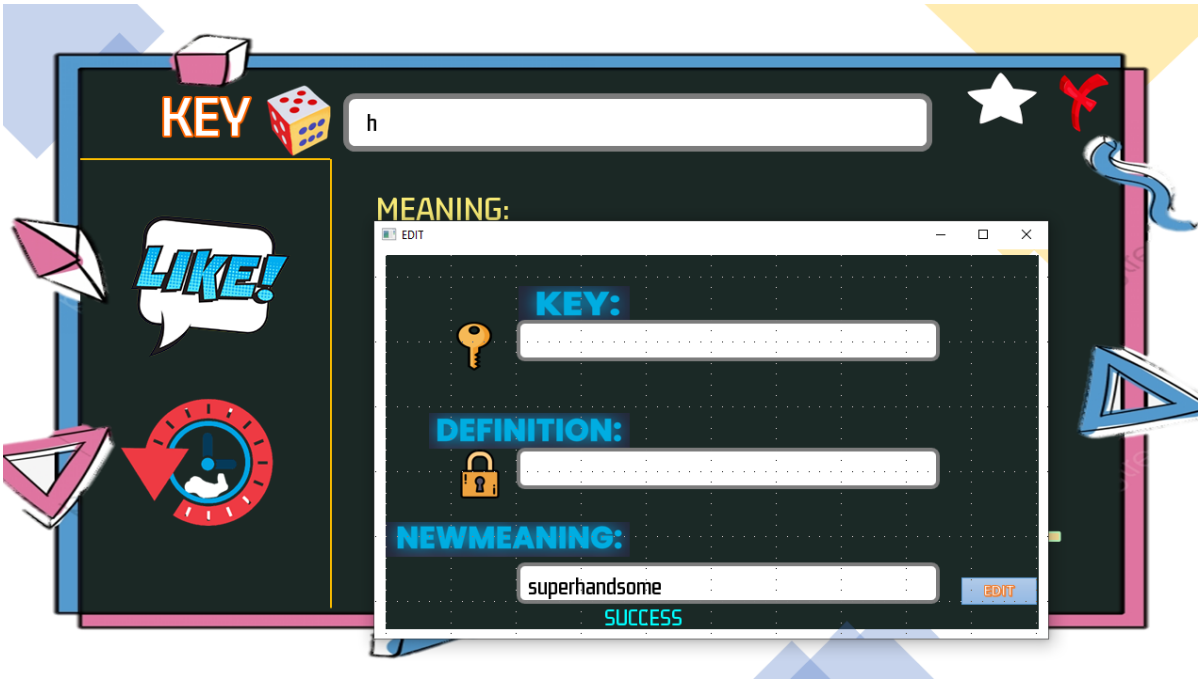
All the looked-up words will be displayed in this window. There will be 2 arrows in this window:

+ **Left arrow**: move backward to previous looked-up words
+ **Right arrow**: move forward to next looked-up words
+ **Delete**: Reset all the history to empty

7. Editing Meaning Page

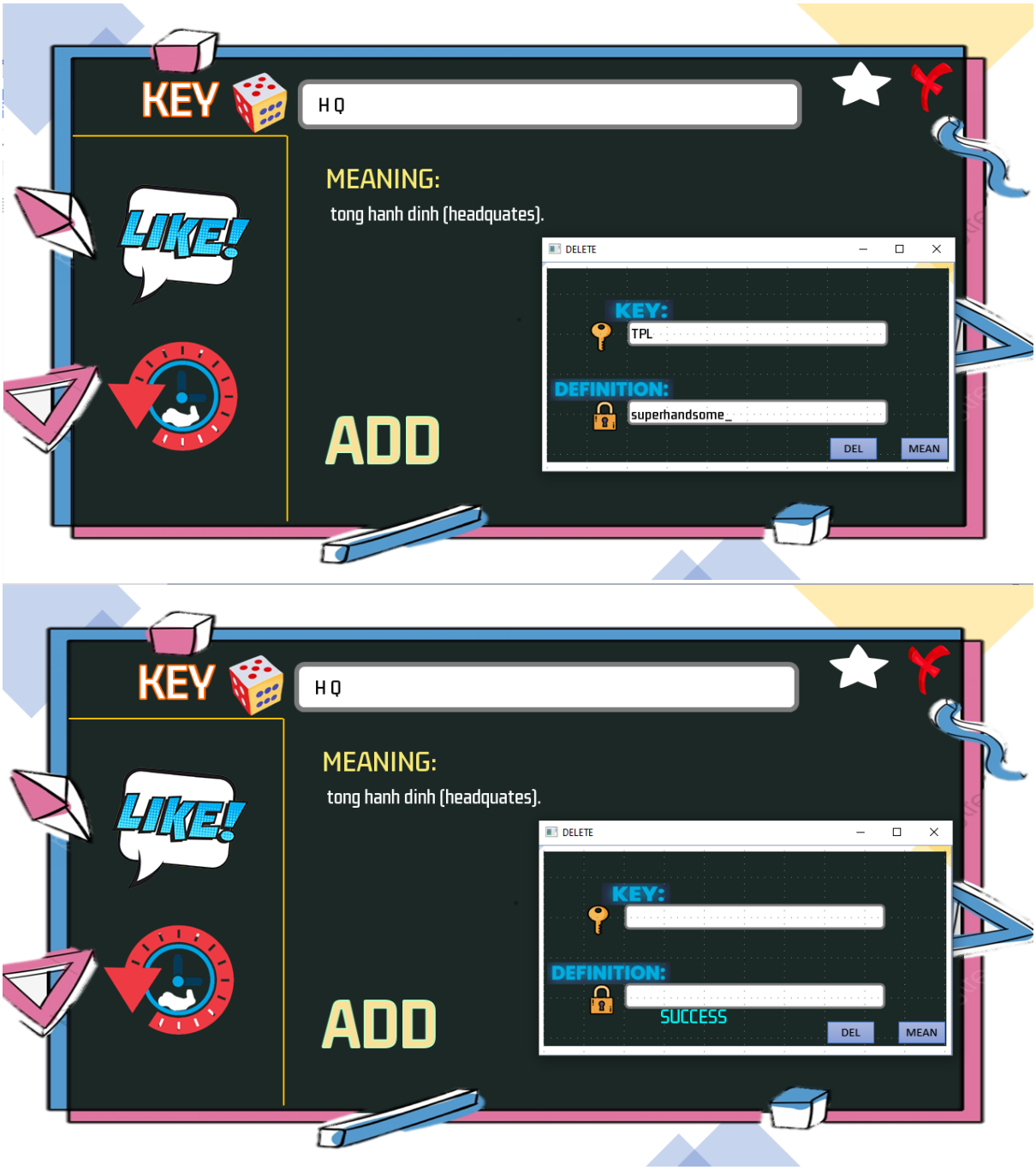


- **Key**: type key that need editing meaning
- **Definition**: type old definition that need correction
- **NewMeaning**: type new meaning that replaces the old one



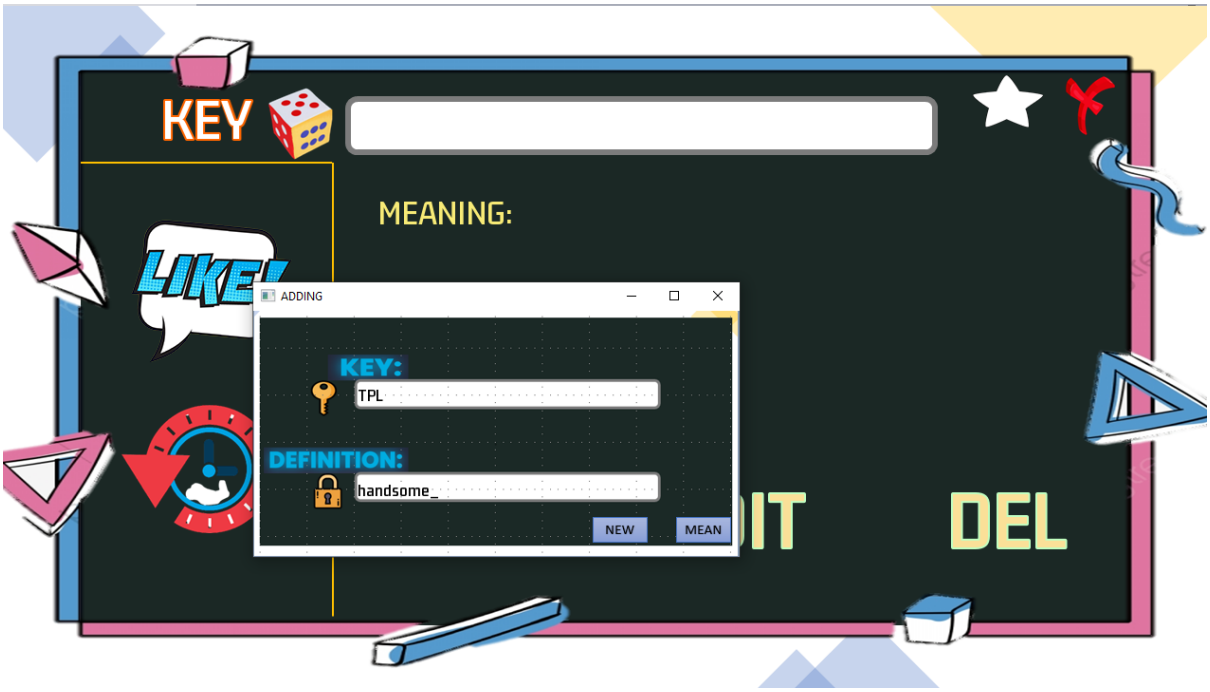
(Display successful window)

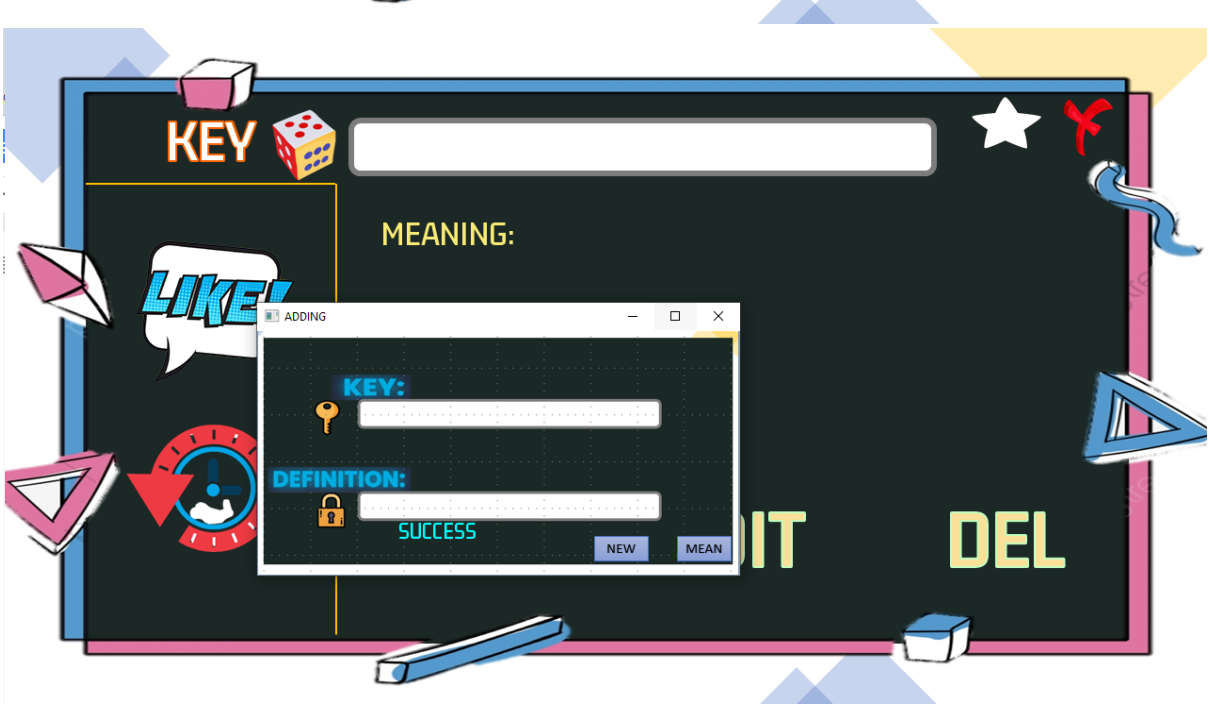
8. Delete Word Page



Type the information and click the **DEL**(delete the word) or **MEAN**(delete the meaning of word)

9. Adding Word Page





Type the information and click **NEW** (add new word) or **MEAN** (add new meaning of word)

10. List of functions done:

1. Users can switch between data sets
2. Users can search for a keyword.
3. Users can add the word to the favorite list to view again later.
4. Users can search for a definition.

5. Users can view the history of search words again.
6. Users can add a new word and its definition.
7. Users can edit the definition of an existing word.
8. Users can remove a word from the dictionary.
9. Users can reset the dictionary to its original state.
10. Users can view a random word and its definition.
11. Users can view the favorite list.
12. Users can remove/add a word from the favorite list.
13. The app can make random a word with four definitions, and users guess its meaning.
14. The app can provide a random definition with four keywords, and users choose the correct word

II. TECHNICAL ANALYSIS

1. Data Structure

- To implement functions for such an application, we have decided to implement a trie. Trie (also known as prefix tree) is an information retrieval data structure used for locating specific keys from within a set. Each branch of a trie represents a possible key among the set, and the corresponding value(s) of the key is stored at the ending node of the branch representing the last character of the key.
- By using a trie, we can optimize the insertion, removal and searching the key down to , with being the length of the key we are dealing with.

2. Functions

- **Adding and searching for a definition/word:**

o To add/search for a word, we will implement the function in a “depth first search” sense. In other words, to check for a word we shall traverse on the trie from the root down to the last node we can move to (the ending node of the key or the last node we can possibly move down to when the key does not exist in the set).

o When adding a new word or the word is an extension of any existing word (the new word and the original word share the same prefix), we should construct the new non-existing nodes of the key to add the key into the trie. Due to such mechanism and the fact that trie is a tree data structure (which means that it is

dynamic in terms of size), it can be deducted that there is no need to deconstruct the entire data structure and the reconstruct it to add the new key.

o Time complexity: O(*len*) with *len* being the length of the word we are adding/searching for.

- **-Removing a word from the dictionary:**

o To be able to remove a word from the trie, we must check whether the word is in the trie or not.

o Basically, the implementation of removing a word from trie is still built upon the depth first search mechanism with a few modifications to deal with specific scenarios:

§ If the word is a unique key (the word does not share with any other words a common prefix), delete all the nodes.

§ If the word is the prefix of another word, remove the leaf node mark.

§ If the word share a common prefix with another word (but not the entire word), delete from the leaf nodes to the “intersection” – the last node of the longest common prefix of the words.

o Time complexity: O(*len*) with *len* being the length of the word we are deleting from the dictionary.

- **Adding a word to the History list:**

o Using a vector to store the list of viewed words.

o Every time the user search for a word/definition, the program should automatically store the word into the vector.

o To view the list, just access into the vector and then check the words.

o Time complexity: O(*1*)

- **Editting definition:**

o Checking if there is this word in the trie.

o Checking if there exist this old meaning of this word in trie.

o Delete this old meaning of this word in trie key-def.

- If in trie def-key, the old meaning does not have any key, delete this old meaning out of trie def-key.

- Adding new meaning for this key through function add new key-def. Because this key existed in trie, this function only push new meaning into vector meaning.

- Adding new meaning and corresponding key into trie def-key.

- Time complexity: O(*3*m+n+k*) with m is the length of the key, n is the length of old meaning, k is the length of the new meaning.

- **Random a word and its meaning:**

- We start random on trie and move using DFS. If the node is not end of word, we start to random which node we should move next.

- If this node is end of word and there is no any next character from that node, choosing this word and its meanings.

- If this node is end of word and there are some next characters from that node, random 0-1 to decide to choose this word or not. If we do not choose this word, start to random which node we should move next and this next node must have character.

- **Random a word and four meanings for game:**

- Firstly, we randomly choose a word and its vector meanings using the function above.

- We randomly choose a meaning of this word in this vector meanings to consider it as right meaning, saving it into a vector four-meaning.

- We start to random on trie def-key to choose a def and its vector keys. We check if the first key exist in this vector keys or not. If it exist, ignore this def and repeat random operation. If this key does not exist in vector keys, check if this def exist in vector four-meaning or not. If it does not exist, choose this def and push it into vector four-meaning.

- We will repeat the step above until my vector four-meaning have 4 elements, including one right meaning of the first key and 3 wrong meaning.

- **Random a meaning and four keys for game:**

- We carry out it similar to the function Random a word and four meanings for game

- **Reset dictionary:**

- Firstly, we save the dictionary into to file, original file of this dictionary and update file. When opening the program, we know that we have two tries to store data key-def and def-key, these tríes store data from update file when opening program. In order to carry out reset operation, my team have also created two original tries and store data from original file. When we choose reset operation, we will deallocate two tries in update-version and copy from original tries to update-version tries.

- Time complexity: reset operation will delete update-version tries and copy original tries so it take O(2*length of update-version tries + 2*length of original tries).

- **Adding a word to the Favorite list:**

- Using a vector to store the list of viewed words.

- Every time the user adds a word to their favorite list, the program should automatically store the word into the vector.

- To view the list, just access into the vector and then check the words.

- Time complexity: O(*1*)

III. CONTRIBUTIONS

	Công việc đã làm
Nguyễn Đức Hưng (2112042)	• Users can add the word to the favorite list to view again later. • Users can view the favorite list. • Users can remove/add a word from the favorite list. • Do UX/UI • 27%

Trần Thiên Phúc (21125094)	• Users can search for a keyword. • Users can search for a definition. • Users can add a new word and its definition. • Users can remove a word from the dictionary. • Design the data structures • 25%
Diệp Tường Nghiêm (21125154)	• Users can edit the definition of an existing word. • Users can view a random word and its definition. • The app can make random a word with four definitions, and users guess its meaning. • The app can provide a random definition with four keywords, and users choose the correct word. • Preprocessing datasets • 25%
Nguyễn Trọng Nghĩa (21125154)	• Users can switch between data sets. • Users can view the history of search words again. • Users can reset the dictionary to its original state • Finding datasets • 23%