📘

CS163 Technical Report, APCS1 - Group08

A technical report explaining our design and algorithms when loading a dataset, searching, adding, updating, or deleting a word; evaluating the O notation of every action, and report the running time.

# 👉 Base Structures

In our product, every dictionary is built under the type name of class Dictionary. Class Dictionary consists of the following members:

* Directory of data file
* 3 vectors storing all Definitions, Words and history, respectively. Definition and Word are defined as C++ classes, and will be explained below.
* A trie data structure for storing and searching words. Class Trie will be discussed below.
* A trie data structure for storing and searching definitions.
* Auxiliaries functions.

## Class Word

We first define a dictionary’s most basic data, words, by creating a C++ class Words whose members are:

* A std::string to store the spelling of that Word.
* An integer storing the index of the Word in Dictionary::allWords vector.
* A pointer of class Definition pointing to its definition(s).
* UI’s auxiliary data.
* A copy constructor.

There is also a class called ResourceWord defined in our project which is used for searching Word-to-Def. This class’ data is the same as class Word, there are only minor adding/removing of members and will not be discussed here.

## Class Definition

Next we built a class representing the next significant data of a dictionary, the definitions, as the class Definition. Definition’s members includes:

* A std::string storing the definition it represents.
* A pointer of class Word pointing to the Word that has this Definition as one of its definitions. Now we conclude the relations of Words and Definitions, Word-Definitions is a one-to-many relation while the reversal is one-to-one. This is true because realistically, a word may have multiple definitions defined in different contexts.
* Back End’s auxiliary data
* A copy constructor

## Trie Structure

* The base unit of our Trie is the Trie\_Node class. Trie\_Node is defined under a template <Record>, which is then used to pass class Word/Resource Word, with these members below:
  + A data whose type is detected by Record.
  + An array of pointers to Record(s) called “branch”. This member defines the path(s) to any possible Record, and is later used in storing and searching.
* Trie Structure is created based on the classic Trie. with Trie\_Node(s) as nodes of the tree structure and also has the same template <Record>. Our Trie has following data members:
  + A char array which stores the path indices of allowed characters.
  + An integer branchLimit limiting the maximum number of branches one node may have.
  + A defaultValue for Record, representing null status.
  + A Trie\_Node pointer *root*, is the root of the tree structure
* Along with various auxiliary functions used for UI/BE/Both. And we only discuss the insert and delete function.
* Inserting
  + To insert a new data to the Trie, we use this function



* + If new\_Data has yet appeared in the Trie, a new Trie\_Node with data = new\_Data is inserted to Trie and return success. Otherwise, return duplicated\_error.
  + The function loops through the parameter newData, getting the integer value of each character to direct the path. If a character is not allowed, return invalid\_character error. If a path exists, move to the next character of newData, else create a new node at that position.
  + At the end, if the destination node is still defaultValue, assign newData to that node’s data. Otherwise, return duplicated\_error.
  + This function uses 1 loop through the string newData, therefore its complexity is linear with the length of newData, which is O(n).
  + Other functions involving traversing the Trie such as find(), search(), trie\_delete()..., just like Inserting, are all linear with the input string parameter’s length. Therefore, they all are of O(n) complexity.

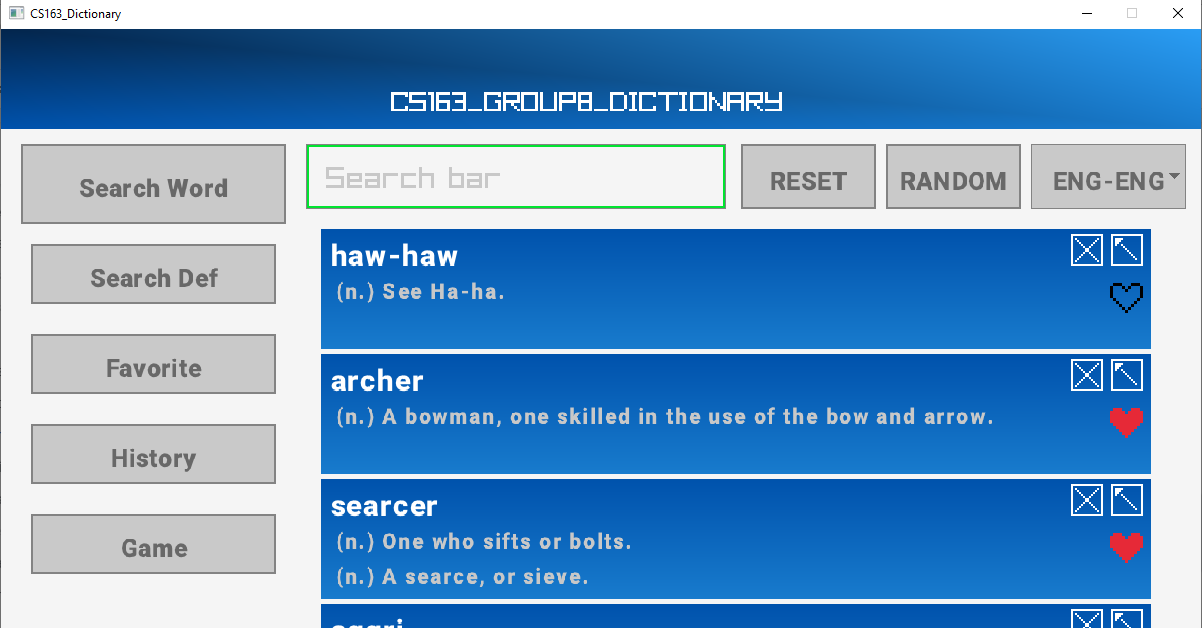
# 👉 Design and algorithms

## Loading a dataset

The Dictionary loads every dataset by creating a separate Trie, and switching to the appropriate trie for the needed dataset. Review “Trie structure” above for deeper reading.

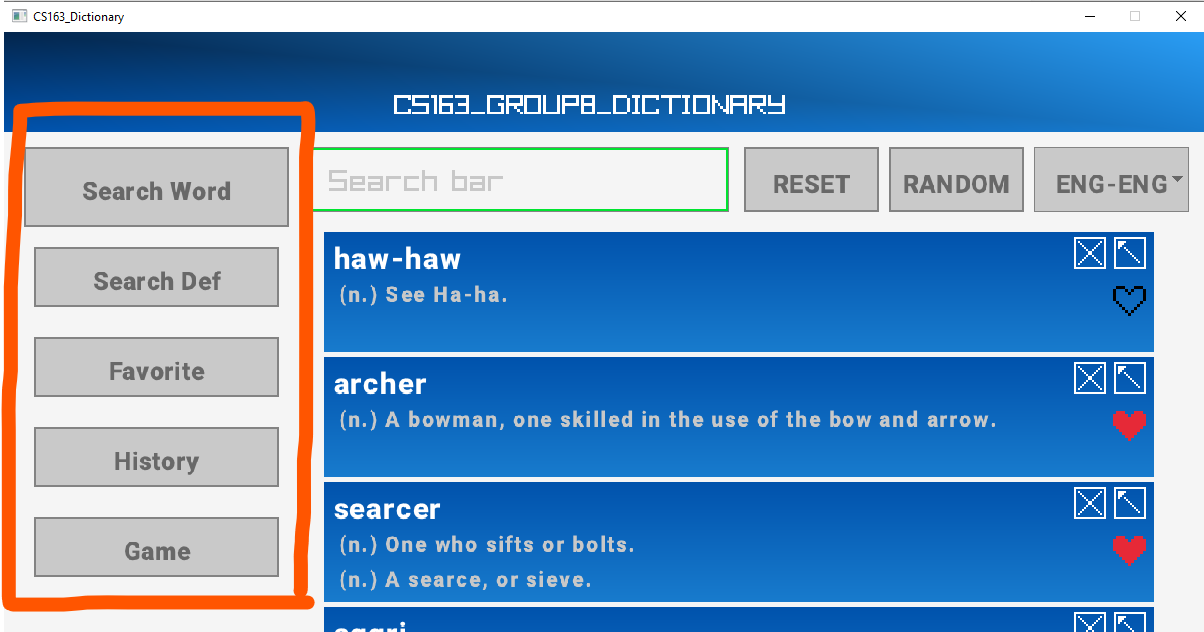
## Searching a word

The most basic functionality of a Dictionary is to search for a word and see its definition(s).Upon launch, our product’s default mode is to search a Word for definitions. Here is our Word-to-Defs searching screen.



*Search Word screen*

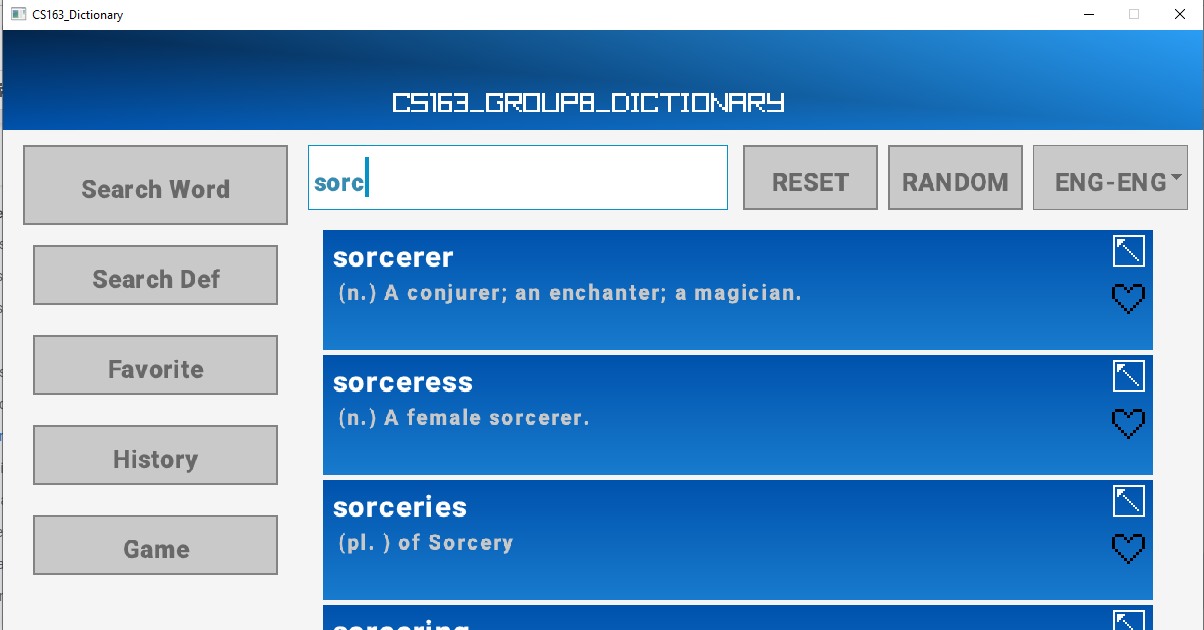
There are several features in Search Word Screen:



This is the Modes Bar, from which your can choose to

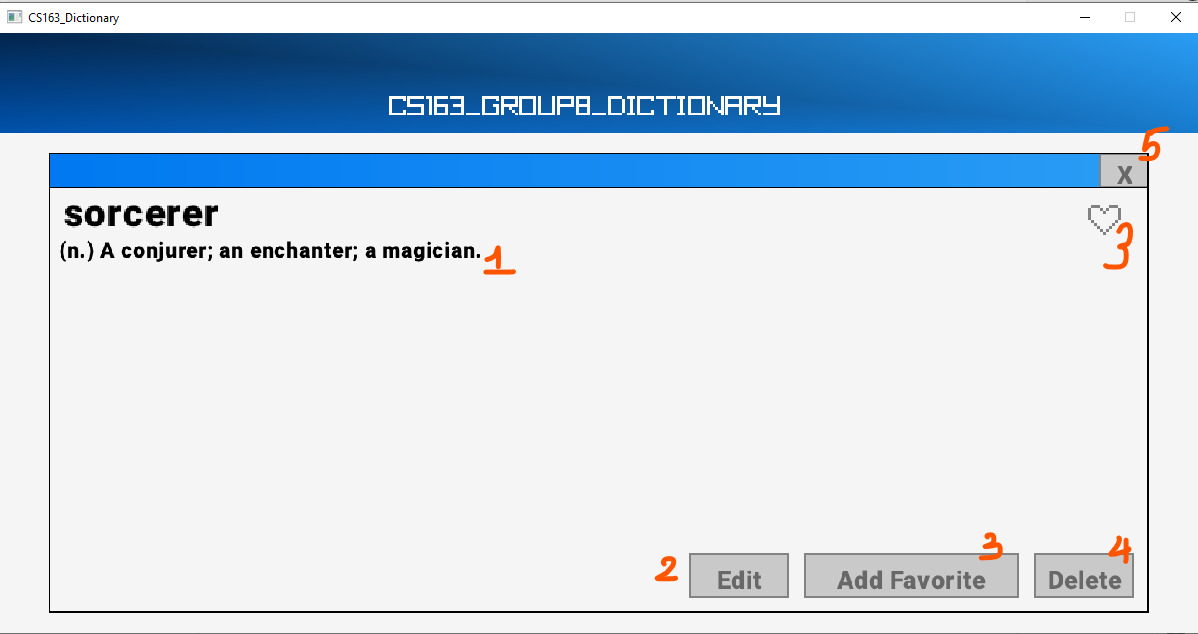
* Change to Def-to-Word search mode (“Search Def”)
* See the list of your favorite words (“Favorite”)
* See the latest searched words (“History”)
* Test your knowledge by trying our Game (“Game”)

To search for a Word, type it in the Search Bar. All possible results shall appear below and you can scroll through them and look for the word you’re seeking for.

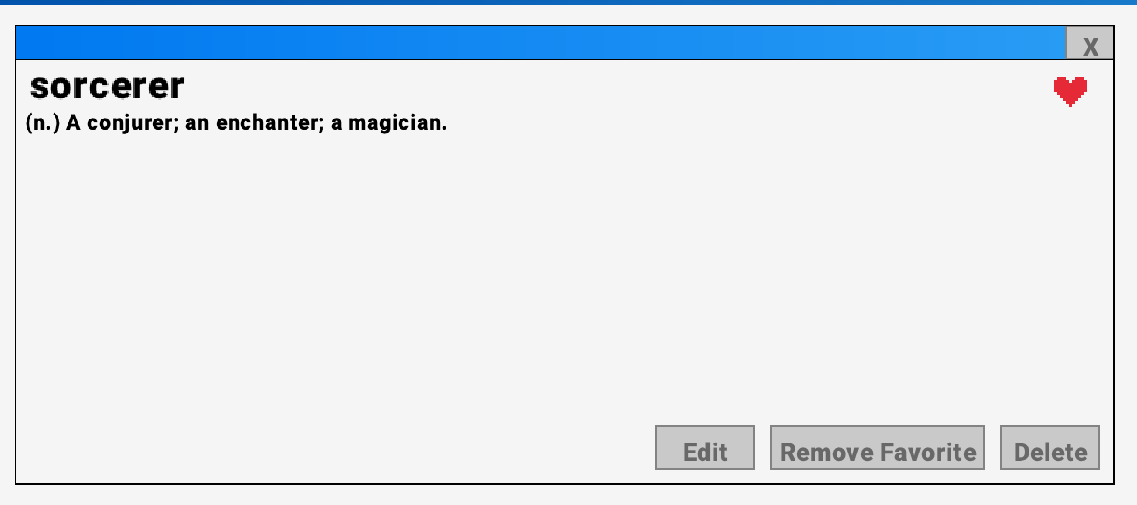


Searching for string “sorc”

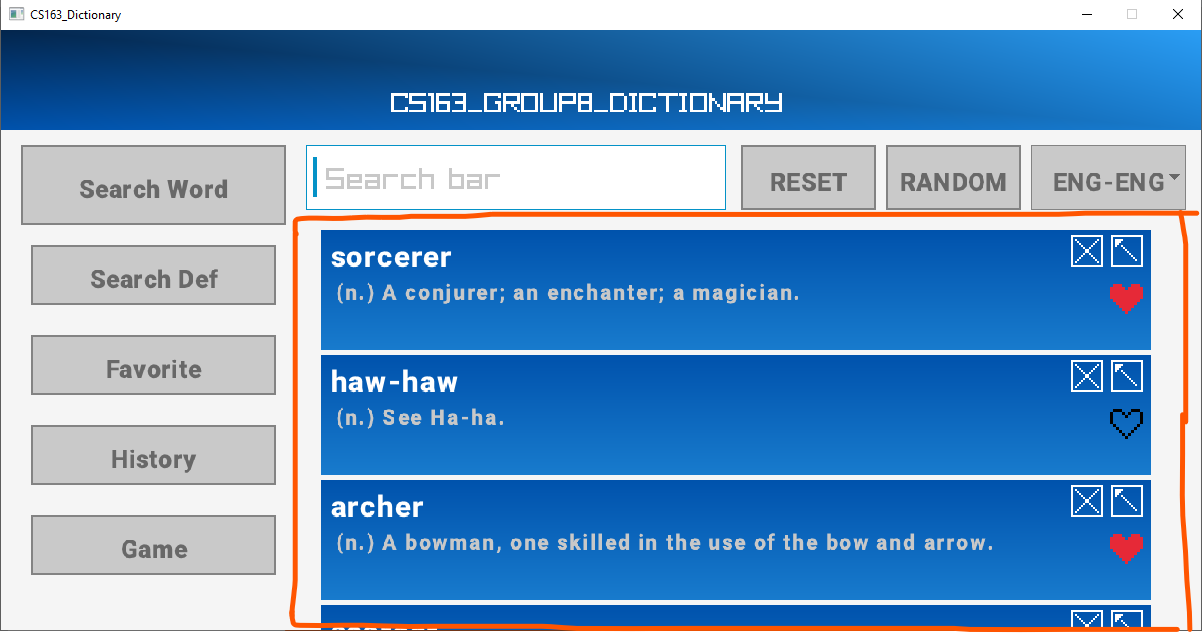
You can choose any result to see it’s detailed information and other options in Word Details screen



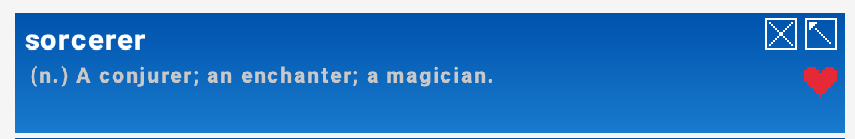
1. The definition(s) of the chosen word
2. Click here to edit the word, you can add/remove definitions from this word, this changes the data of the Dictionary.
3. Add this word to the Favorite list, the heart represents the Favorite status (white and red means “is not a favorite” and “is a favorite”, respectively)
4. Remove this word from the Dictionary, this changes the data of the Dictionary
5. Go back to the main screen



*Added “sorcerer” to Favorite list, Heart is now Red indicating sorcerer is one of your favorites*



If there is no input to Search Bar, the list below shall display the History of previous searchings.

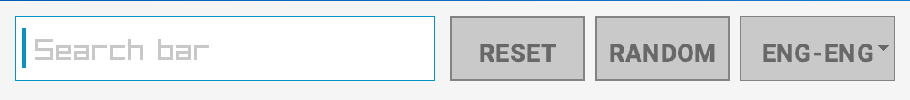


In the searching result list or history list displayed below the Search Bar, these results called Result Boxes. Each box has a brief information of a word that might be a possible result or a recently searched word (depends on whether you are searching or not) and 2 buttons on the upper right corner.

The X button  removes this result/history from the display list

The left-upward arrow button  quickly fill the Search Bar with the word in this Result Box, save your time from typing manually.

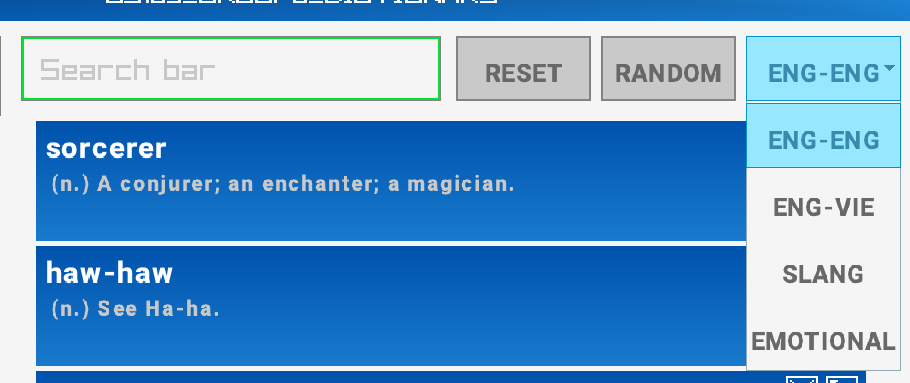
Also in the Word-to-Defs searching menu, next to the Search Bar is 3 options:



“RESET”: reset the data of the Dictionary to the original data set, reverting all changes user made.

“RANDOM”: randomly choose a word from the list of all words, search for that word.

“ENG-ENG”: This button displays the current type of dictionary in use, the default is the Eng-Eng dictionary. You can click on this button to change to other types our product provides.



*Choose another dictionary.*

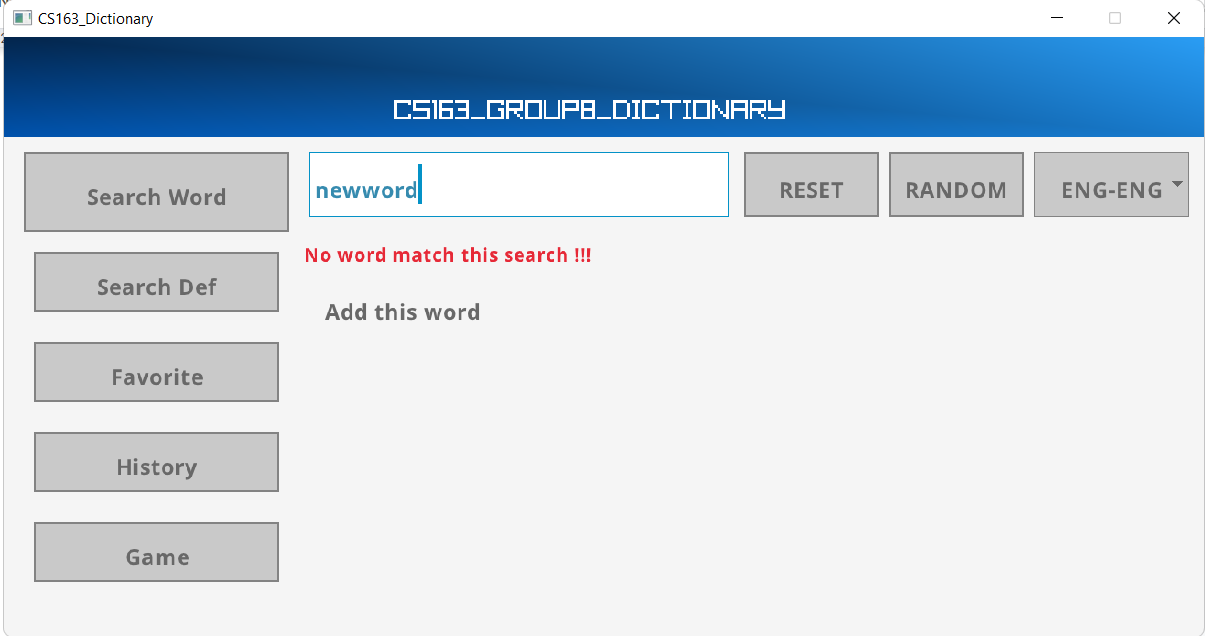
## Searching a Definition

Def-to-Word searching is used just like Word-to-Defs functionality. You can input any string that fits the definition of the word you are looking up, press Enter when finished and every word whose one/many definitions resembles the input string will appear below the Search Bar.

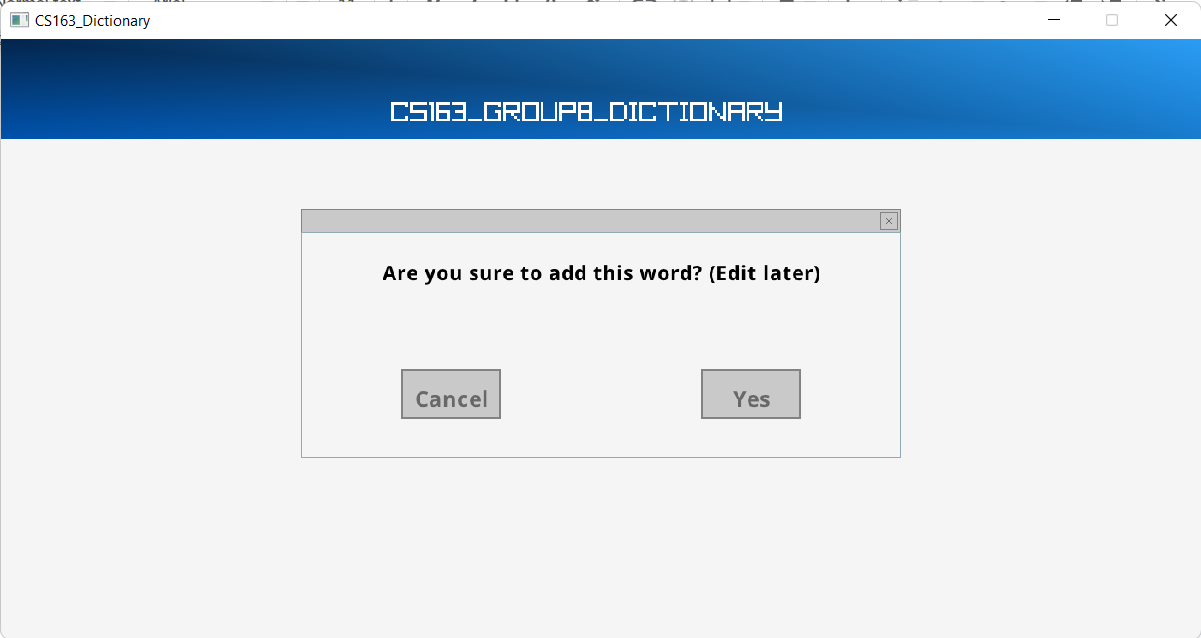


## Adding a word

When you search a word that have not in the dictionary yet, there will be an announcement and a button for adding a word like this.



If you click the button, a box will be appear to make sure you want to add this word

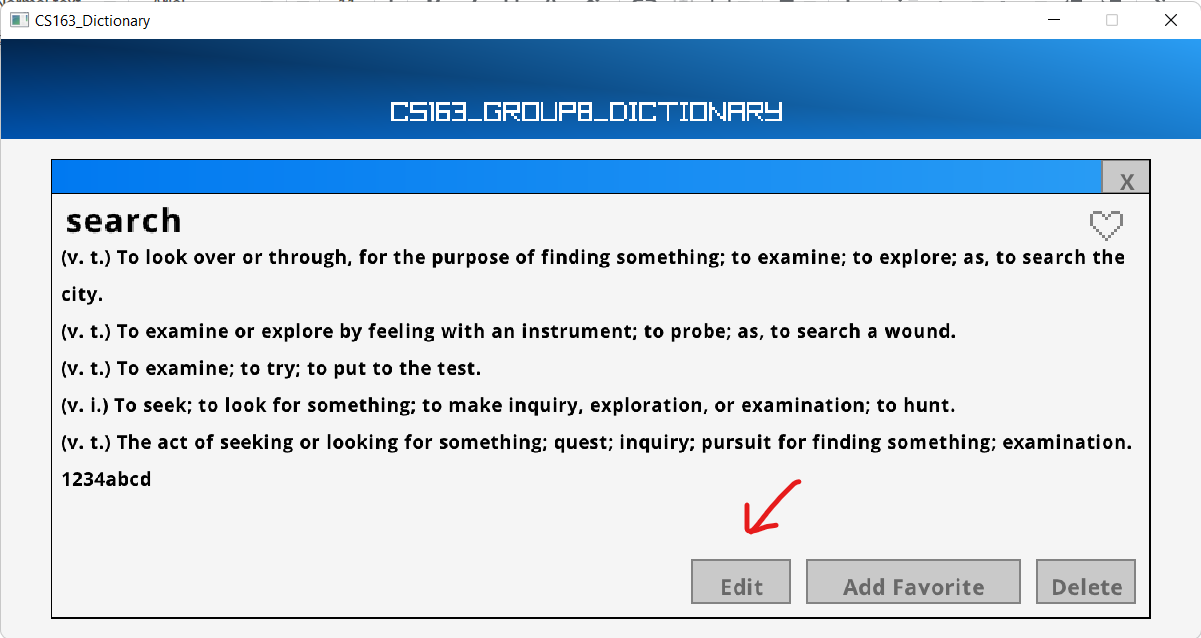


Then, choosing “Yes” to add the word and you can find out, edit it later.

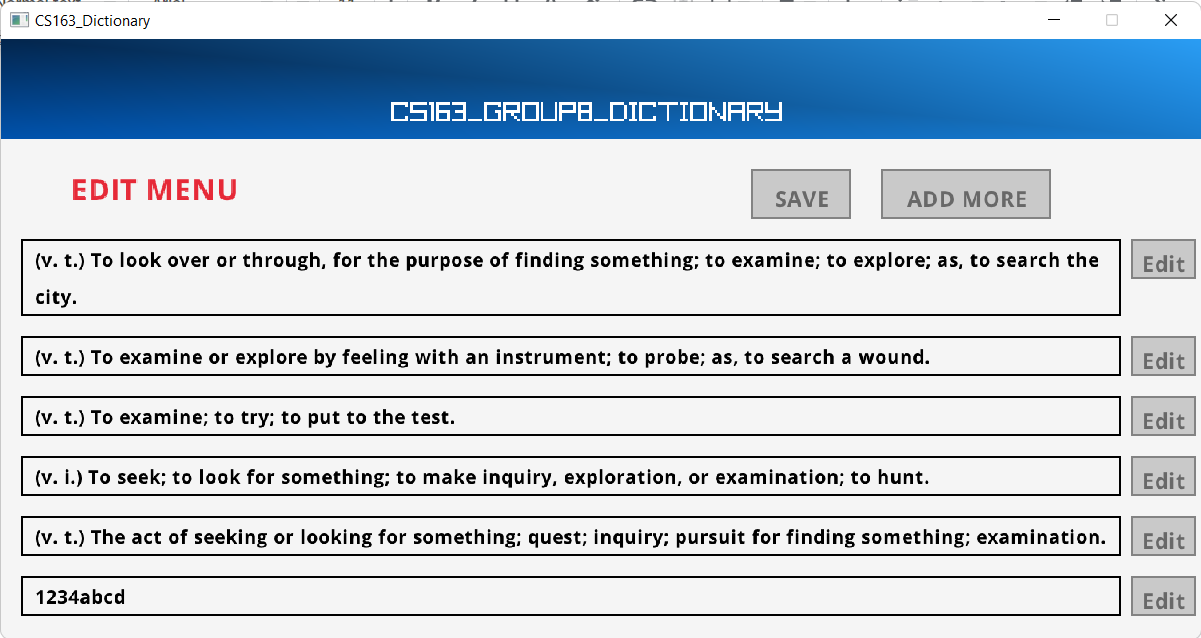
## Updating a word

In definition menu of word, you can choose:

* Edit: To edit definition



Then, a menu for you to edit appear and you can edit anything you want



After editing, you click “SAVE” button to save the changes.

## Deleting a word

Also in definition menu, you can delete the word when click “Delete” button.

