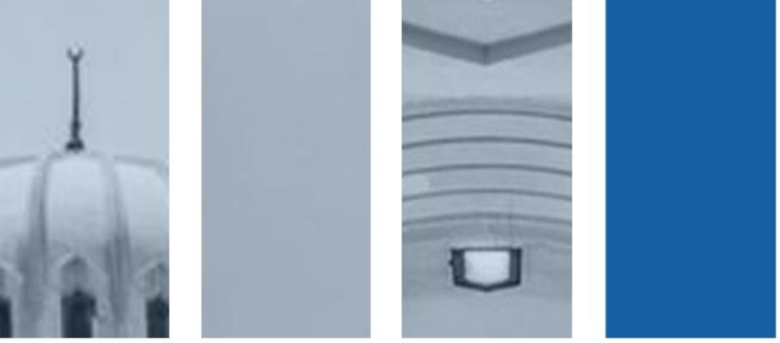


Project II: Inverted index

Program:

Course Code: CSE323

Course Name: Data structures

Examination Committee

Dr. Islam El-Maddah

Ain Shams University
Faculty of Engineering
Spring Semester – 2020

Student Personal Information for Group Work

Student Names:	Student Codes:
Abldelrahman Atef :	1600790
Omar Ahmed :	1600851
Shimaa Mohamed :	1600703
Omar Hatem :	1600865
Doaa Hesham :	1600524

Plagiarism Statement

I certify that this assignment / report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment / report has not been previously been submitted for assessment for another course. I certify that I have not copied in part or whole or otherwise plagiarized the work of other students and / or persons.

Signature/Student Name: **Omar Ahmed**
Abdelrahman Atef
Shimaa Mohamed
Omar Hatem
Doaa Hesham

Date: 20/5/2020

Submission Contents

01

First Topic

Abstract:

this project’s goal is to efficiently build an inverted index, for a database of documents, and then to use the index to get statistics for any word (Ie: how many times this word occurred and in which documents).

Here’s a Video explaining how the project works.

- Video

Here’s the Project's Repository on GitHub.

-Github

02

Second Topic

Background:

Indexing is the first step in any Information retrieval system (IR), it could be normal (forward), or it could be inverted.

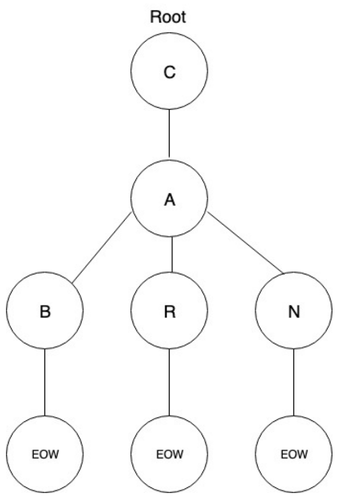
Inverted Index :

Implementation: is basically mapping from words to documents, so for each word, we would references to all document that has this word.

Forward Indexing: is basically mapping from document to words, so for each document we index some keywords . And when we search for a keywords the result would be documents that have this keyword. Example of Search Engine that uses forward indexing is Xapian, which is an Open Source Search Engine Library.

In this project, we implemented an inverted index system. To build it, we could either use a binary search tree (BST), where each node would be a word and list of files that have this word, but the complexity of inserting a word, and for searching for one would be O(log h), where h is the tree's height. Another option is to use Trie, which we did, as it's more efficient memory and time wise.

A trie is a tree data structure, where each node for a letter, or end of word node that contains list of all documents that has this word. So now the complexity on inserting or searching a word is the length of the word itself, which is better than using BST.



Sample of a Trie, Src

we also used Json to store our indexing database, so we needed to parse data from the trie to Json format, and vice versa.

Json is short for “JavaScript Object Notation”, it’s basically a key-value pair file format.

C++ doesn't natively support Json, so we used json.hpp library by nlohmann. It’s a C++ lib that facilitates dealing with json format for C++ projects.

We stored the indexing trie in a josn file for each Input folder. We stored it in a key-value pair, where the key is the word, and the value is a list of all files’ paths that contains this word.

Then we would retrieve data from this Json file to retrieve the query results for any word.

This is a graphical user interface (GUI) application, so for the GUI part, we used wxWidgets .

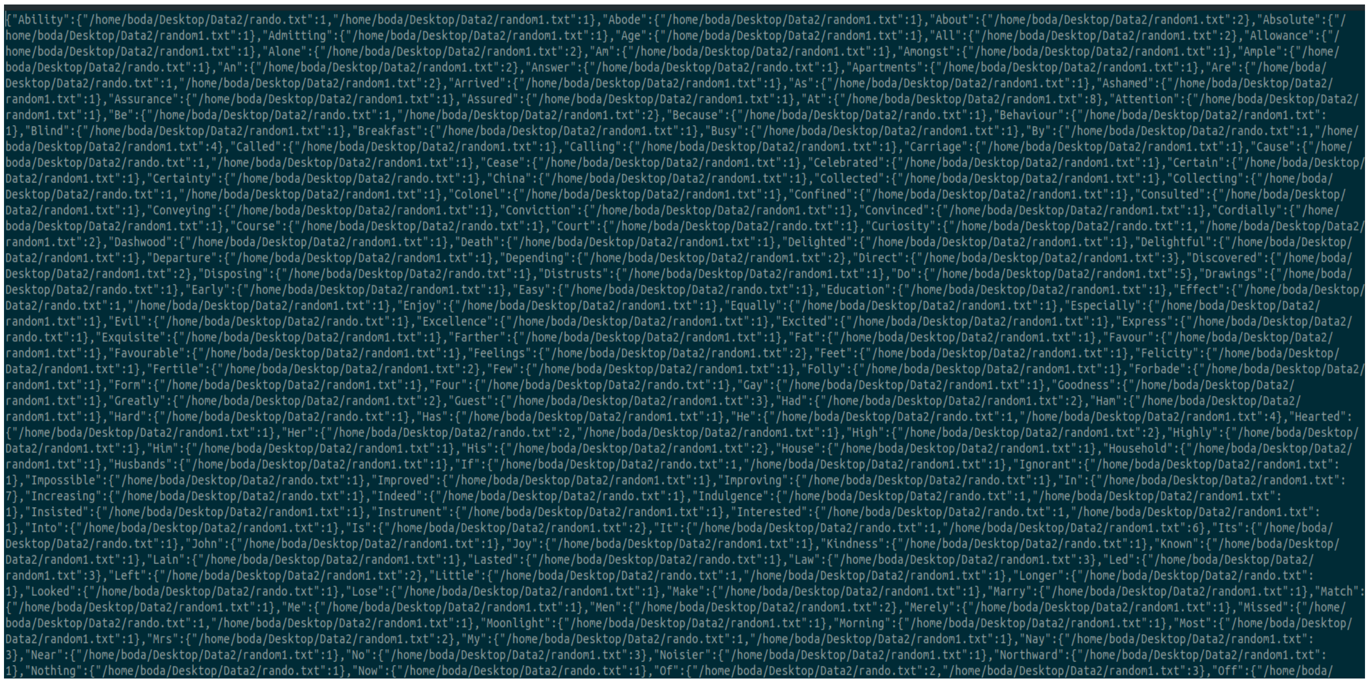
wxWidgets is a C++ library that lets developers create applications for Windows, macOS, Linux and other platforms with a single code base.

How our Project works:

When the program runs for the first time, it would create a folder in the same folder of the binary file, and name it “Database”, which would work as the database for the project.

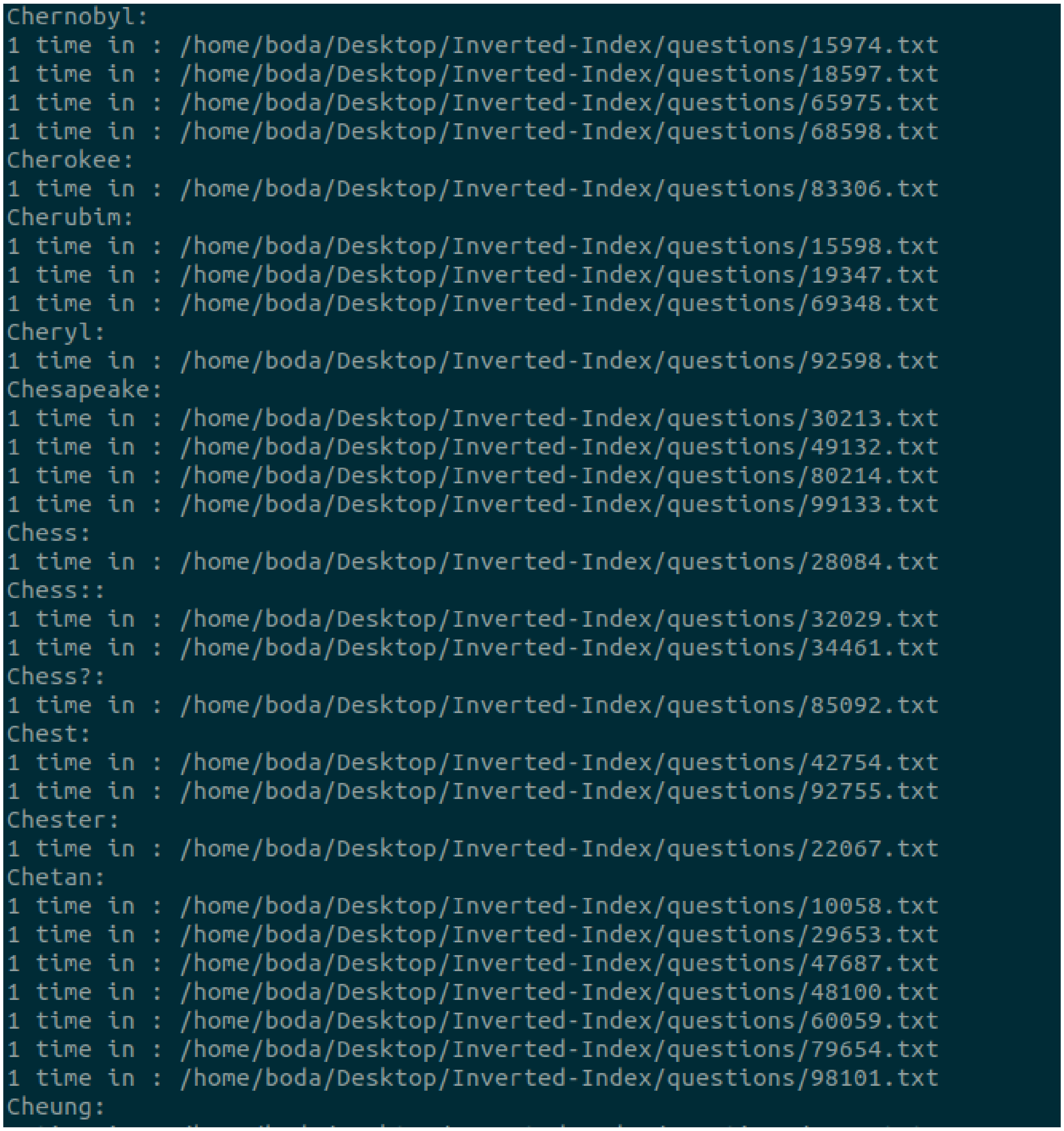
After indexing a folder, two new files would be added to the Database folder, a Json file, and a txt file.

the json file is used to store the indexing, and retrieve it again.



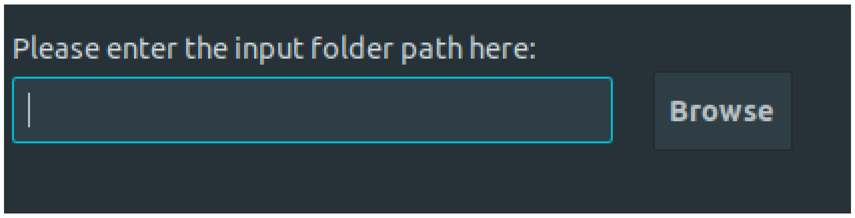
```
{"Ability":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Abode":{"/home/boda/Desktop/Data2/random1.txt":1},"About":{"/home/boda/Desktop/Data2/random1.txt":2},"Absolute":{"/
home/boda/Desktop/Data2/random1.txt":1},"Admitting":{"/home/boda/Desktop/Data2/random1.txt":1},"Age":{"/home/boda/Desktop/Data2/random1.txt":1},"All":{"/home/boda/Desktop/Data2/random1.txt":2},"Allowance":{"/
home/boda/Desktop/Data2/random1.txt":1},"Alone":{"/home/boda/Desktop/Data2/random1.txt":2},"Am":{"/home/boda/Desktop/Data2/random1.txt":1},"Amongst":{"/home/boda/Desktop/Data2/random1.txt":1},"Ample":{"/home/
boda/Desktop/Data2/rando.txt":1},"An":{"/home/boda/Desktop/Data2/random1.txt":2},"Answer":{"/home/boda/Desktop/Data2/rando.txt":1},"Apartments":{"/home/boda/Desktop/Data2/random1.txt":1},"Are":{"/home/boda/
Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":2},"Arrived":{"/home/boda/Desktop/Data2/random1.txt":1},"As":{"/home/boda/Desktop/Data2/random1.txt":1},"Ashamed":{"/home/boda/Desktop/Data2/
random1.txt":1},"Assurance":{"/home/boda/Desktop/Data2/random1.txt":1},"Assured":{"/home/boda/Desktop/Data2/random1.txt":1},"At":{"/home/boda/Desktop/Data2/random1.txt":8},"Attention":{"/home/boda/Desktop/Data2/
random1.txt":1},"Be":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":2},"Because":{"/home/boda/Desktop/Data2/rando.txt":1},"Behaviour":{"/home/boda/Desktop/Data2/random1.txt":
1},"Blind":{"/home/boda/Desktop/Data2/random1.txt":1},"Breakfast":{"/home/boda/Desktop/Data2/random1.txt":1},"Busy":{"/home/boda/Desktop/Data2/random1.txt":1},"By":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/
boda/Desktop/Data2/random1.txt":4},"Called":{"/home/boda/Desktop/Data2/random1.txt":1},"Calling":{"/home/boda/Desktop/Data2/random1.txt":1},"Carriage":{"/home/boda/Desktop/Data2/random1.txt":1},"Cause":{"/home/
boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Cease":{"/home/boda/Desktop/Data2/random1.txt":1},"Celebrated":{"/home/boda/Desktop/Data2/random1.txt":1},"Certain":{"/home/boda/Desktop/
Data2/random1.txt":1},"Certainty":{"/home/boda/Desktop/Data2/rando.txt":1},"China":{"/home/boda/Desktop/Data2/random1.txt":1},"Collected":{"/home/boda/Desktop/Data2/random1.txt":1},"Collecting":{"/home/boda/
Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Colonel":{"/home/boda/Desktop/Data2/random1.txt":1},"Confined":{"/home/boda/Desktop/Data2/random1.txt":1},"Consulted":{"/home/boda/Desktop/
Data2/random1.txt":1},"Conveying":{"/home/boda/Desktop/Data2/random1.txt":1},"Conviction":{"/home/boda/Desktop/Data2/random1.txt":1},"Convinced":{"/home/boda/Desktop/Data2/random1.txt":1},"Cordially":{"/home/
boda/Desktop/Data2/random1.txt":1},"Course":{"/home/boda/Desktop/Data2/rando.txt":1},"Court":{"/home/boda/Desktop/Data2/rando.txt":1},"Curiosity":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/
random1.txt":2},"Dashwood":{"/home/boda/Desktop/Data2/random1.txt":1},"Death":{"/home/boda/Desktop/Data2/random1.txt":1},"Delighted":{"/home/boda/Desktop/Data2/random1.txt":1},"Delightful":{"/home/boda/Desktop/
Data2/random1.txt":1},"Departure":{"/home/boda/Desktop/Data2/random1.txt":1},"Depending":{"/home/boda/Desktop/Data2/random1.txt":2},"Direct":{"/home/boda/Desktop/Data2/random1.txt":3},"Discovered":{"/home/boda/
Desktop/Data2/random1.txt":2},"Disposing":{"/home/boda/Desktop/Data2/rando.txt":1},"Distrusts":{"/home/boda/Desktop/Data2/random1.txt":1},"Do":{"/home/boda/Desktop/Data2/random1.txt":5},"Drawings":{"/home/boda/
Desktop/Data2/rando.txt":1},"Early":{"/home/boda/Desktop/Data2/random1.txt":1},"Easy":{"/home/boda/Desktop/Data2/rando.txt":1},"Education":{"/home/boda/Desktop/Data2/random1.txt":1},"Effect":{"/home/boda/Desktop/
Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Enjoy":{"/home/boda/Desktop/Data2/random1.txt":1},"Equally":{"/home/boda/Desktop/Data2/random1.txt":1},"Especially":{"/home/boda/Desktop/Data2/
random1.txt":1},"Evil":{"/home/boda/Desktop/Data2/rando.txt":1},"Excellence":{"/home/boda/Desktop/Data2/random1.txt":1},"Excited":{"/home/boda/Desktop/Data2/random1.txt":1},"Express":{"/home/boda/Desktop/Data2/
rando.txt":1},"Exquisite":{"/home/boda/Desktop/Data2/random1.txt":1},"Farther":{"/home/boda/Desktop/Data2/random1.txt":1},"Fat":{"/home/boda/Desktop/Data2/random1.txt":1},"Favour":{"/home/boda/Desktop/Data2/
random1.txt":1},"Favourable":{"/home/boda/Desktop/Data2/random1.txt":1},"Feelings":{"/home/boda/Desktop/Data2/random1.txt":2},"Feet":{"/home/boda/Desktop/Data2/random1.txt":1},"Felicity":{"/home/boda/Desktop/
Data2/random1.txt":1},"Fertile":{"/home/boda/Desktop/Data2/random1.txt":2},"Few":{"/home/boda/Desktop/Data2/random1.txt":1},"Folly":{"/home/boda/Desktop/Data2/random1.txt":1},"Forbade":{"/home/boda/Desktop/Data2/
random1.txt":1},"Form":{"/home/boda/Desktop/Data2/random1.txt":1},"Four":{"/home/boda/Desktop/Data2/rando.txt":1},"Gay":{"/home/boda/Desktop/Data2/random1.txt":1},"Goodness":{"/home/boda/Desktop/Data2/
random1.txt":1},"Greatly":{"/home/boda/Desktop/Data2/random1.txt":2},"Guest":{"/home/boda/Desktop/Data2/random1.txt":3},"Had":{"/home/boda/Desktop/Data2/random1.txt":2},"Ham":{"/home/boda/Desktop/Data2/
random1.txt":1},"Hard":{"/home/boda/Desktop/Data2/rando.txt":1},"Has":{"/home/boda/Desktop/Data2/random1.txt":1},"He":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":4},"Hearted":
{"/home/boda/Desktop/Data2/random1.txt":1},"Her":{"/home/boda/Desktop/Data2/rando.txt":2,"/home/boda/Desktop/Data2/random1.txt":1},"High":{"/home/boda/Desktop/Data2/random1.txt":2},"Highly":{"/home/boda/Desktop/
Data2/random1.txt":1},"Him":{"/home/boda/Desktop/Data2/random1.txt":1},"His":{"/home/boda/Desktop/Data2/random1.txt":2},"House":{"/home/boda/Desktop/Data2/random1.txt":1},"Household":{"/home/boda/Desktop/Data2/
random1.txt":1},"Husbands":{"/home/boda/Desktop/Data2/random1.txt":1},"If":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Ignorant":{"/home/boda/Desktop/Data2/random1.txt":
1},"Impossible":{"/home/boda/Desktop/Data2/rando.txt":1},"Improved":{"/home/boda/Desktop/Data2/random1.txt":1},"Improving":{"/home/boda/Desktop/Data2/rando.txt":1},"In":{"/home/boda/Desktop/Data2/random1.txt":
7},"Increasing":{"/home/boda/Desktop/Data2/rando.txt":1},"Indeed":{"/home/boda/Desktop/Data2/rando.txt":1},"Indulgence":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":
1},"Insisted":{"/home/boda/Desktop/Data2/random1.txt":1},"Instrument":{"/home/boda/Desktop/Data2/random1.txt":1},"Interested":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":
1},"Into":{"/home/boda/Desktop/Data2/random1.txt":1},"Is":{"/home/boda/Desktop/Data2/random1.txt":2},"It":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":6},"Its":{"/home/boda/
Desktop/Data2/rando.txt":1},"John":{"/home/boda/Desktop/Data2/random1.txt":1},"Joy":{"/home/boda/Desktop/Data2/random1.txt":1},"Kindness":{"/home/boda/Desktop/Data2/rando.txt":1},"Known":{"/home/boda/Desktop/
Data2/random1.txt":1},"Lain":{"/home/boda/Desktop/Data2/random1.txt":1},"Lasted":{"/home/boda/Desktop/Data2/rando.txt":1},"Law":{"/home/boda/Desktop/Data2/random1.txt":3},"Led":{"/home/boda/Desktop/Data2/
random1.txt":3},"Left":{"/home/boda/Desktop/Data2/random1.txt":2},"Little":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Longer":{"/home/boda/Desktop/Data2/rando.txt":
1},"Looked":{"/home/boda/Desktop/Data2/rando.txt":1},"Lose":{"/home/boda/Desktop/Data2/random1.txt":1},"Make":{"/home/boda/Desktop/Data2/random1.txt":1},"Marry":{"/home/boda/Desktop/Data2/random1.txt":1},"Match":
{"/home/boda/Desktop/Data2/random1.txt":1},"Me":{"/home/boda/Desktop/Data2/random1.txt":1},"Men":{"/home/boda/Desktop/Data2/random1.txt":2},"Merely":{"/home/boda/Desktop/Data2/random1.txt":1},"Missed":{"/home/
boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Moonlight":{"/home/boda/Desktop/Data2/random1.txt":1},"Morning":{"/home/boda/Desktop/Data2/random1.txt":1},"Most":{"/home/boda/Desktop/
Data2/random1.txt":1},"Mrs":{"/home/boda/Desktop/Data2/random1.txt":2},"My":{"/home/boda/Desktop/Data2/rando.txt":1,"/home/boda/Desktop/Data2/random1.txt":1},"Nay":{"/home/boda/Desktop/Data2/random1.txt":
3},"Near":{"/home/boda/Desktop/Data2/random1.txt":1},"No":{"/home/boda/Desktop/Data2/random1.txt":3},"Noisier":{"/home/boda/Desktop/Data2/random1.txt":1},"Northward":{"/home/boda/Desktop/Data2/random1.txt":
1},"Nothing":{"/home/boda/Desktop/Data2/rando.txt":1},"Now":{"/home/boda/Desktop/Data2/rando.txt":1},"Of":{"/home/boda/Desktop/Data2/rando.txt":2,"/home/boda/Desktop/Data2/random1.txt":3},"Off":{"/home/boda/
```

The text file works as a general view for the statistics of the words, it contain all words and the statistics for each word

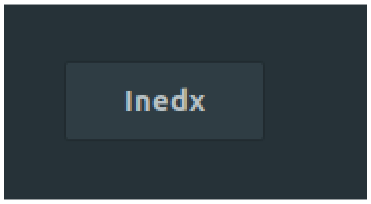


```
Chernobyl:
1 time in : /home/boda/Desktop/Inverted-Index/questions/15974.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/18597.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/65975.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/68598.txt
Cherokee:
1 time in : /home/boda/Desktop/Inverted-Index/questions/83306.txt
Cherubim:
1 time in : /home/boda/Desktop/Inverted-Index/questions/15598.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/19347.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/69348.txt
Cheryl:
1 time in : /home/boda/Desktop/Inverted-Index/questions/92598.txt
Chesapeake:
1 time in : /home/boda/Desktop/Inverted-Index/questions/30213.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/49132.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/80214.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/99133.txt
Chess:
1 time in : /home/boda/Desktop/Inverted-Index/questions/28084.txt
Chess::
1 time in : /home/boda/Desktop/Inverted-Index/questions/32029.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/34461.txt
Chess?:
1 time in : /home/boda/Desktop/Inverted-Index/questions/85092.txt
Chest:
1 time in : /home/boda/Desktop/Inverted-Index/questions/42754.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/92755.txt
Chester:
1 time in : /home/boda/Desktop/Inverted-Index/questions/22067.txt
Chetan:
1 time in : /home/boda/Desktop/Inverted-Index/questions/10058.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/29653.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/47687.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/48100.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/60059.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/79654.txt
1 time in : /home/boda/Desktop/Inverted-Index/questions/98101.txt
Cheung:
```

First the user would have a edit text box to enter the input folder's path manually, or he can press on the "browse" button to choose a file from the file explorer



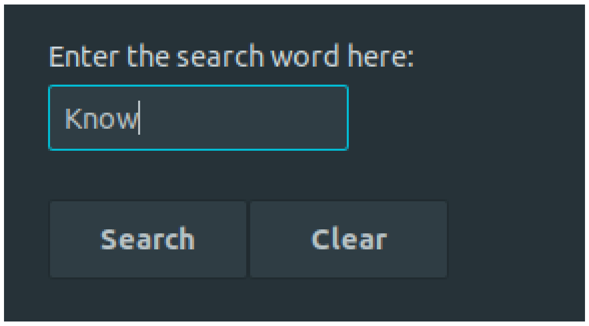
Then he would press "Index" button to index the folder :



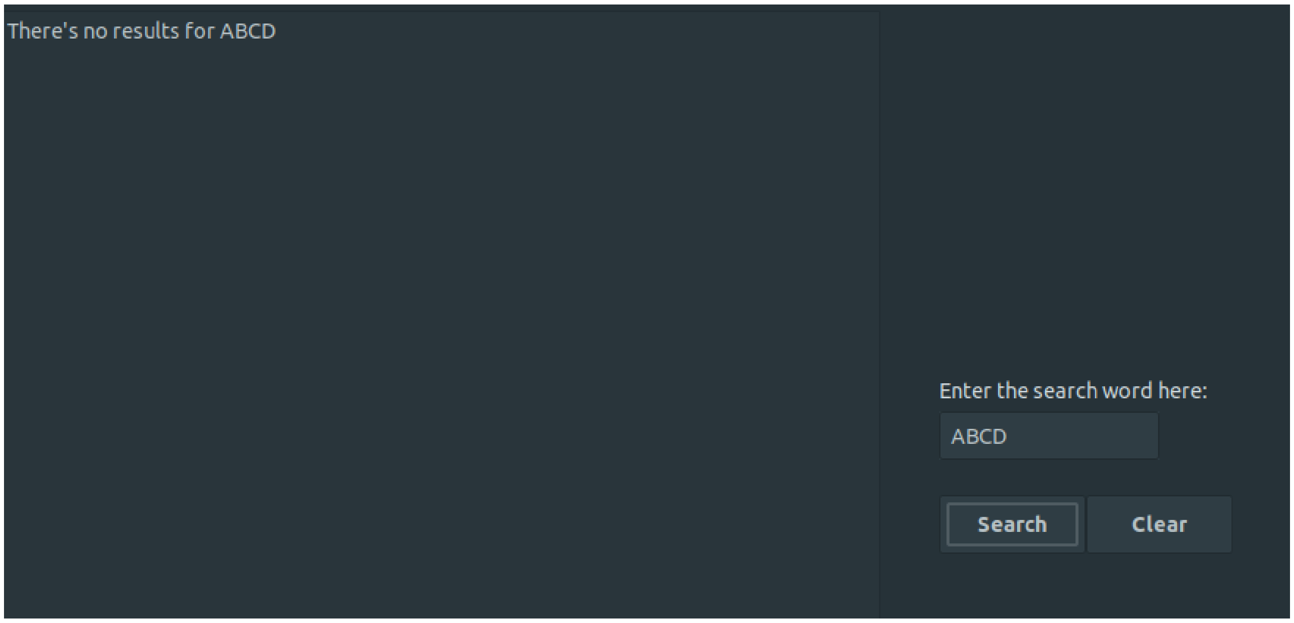
If this folder was indexed before, then the user will see a message telling him that this folder was indexed before, and then no need to do the work twice.

If this is the first time to index this folder, then It would be start indexing it, and I message would appear to user telling him that it finished indexing

Then the user would enter the word he want to search for in the edit text box and press "Search"

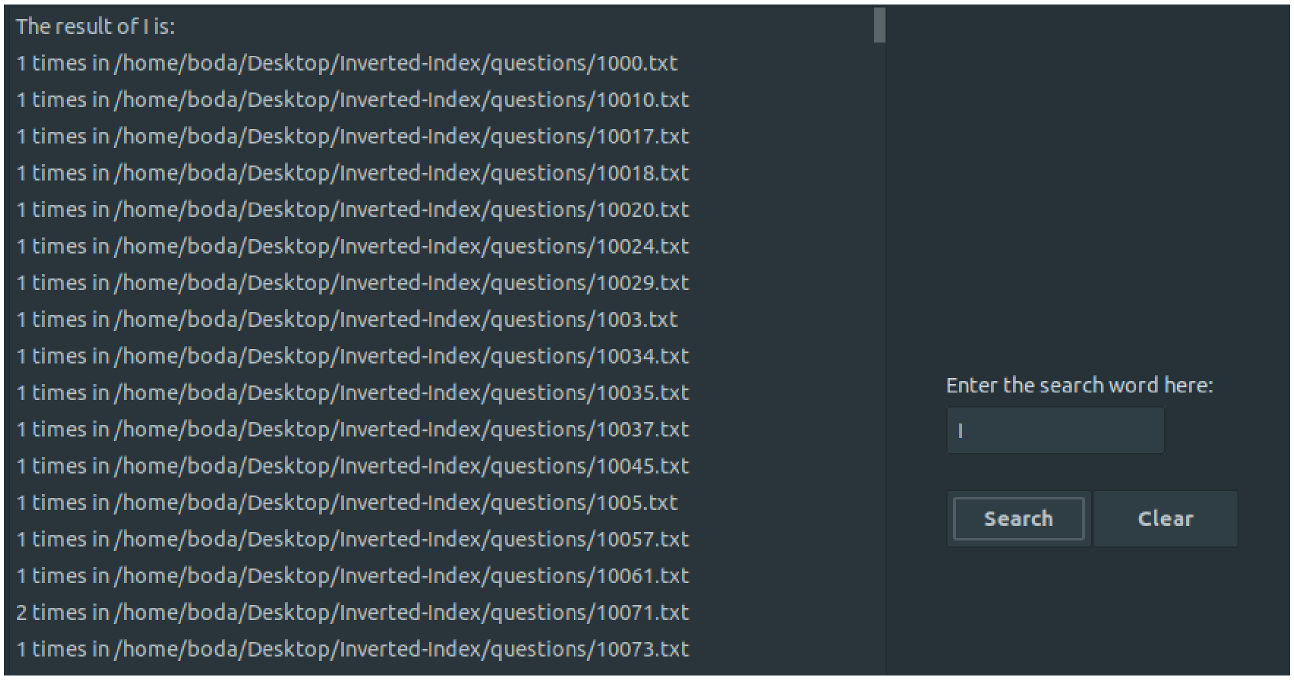


If there's no result for the word, the program would output a message saying so :



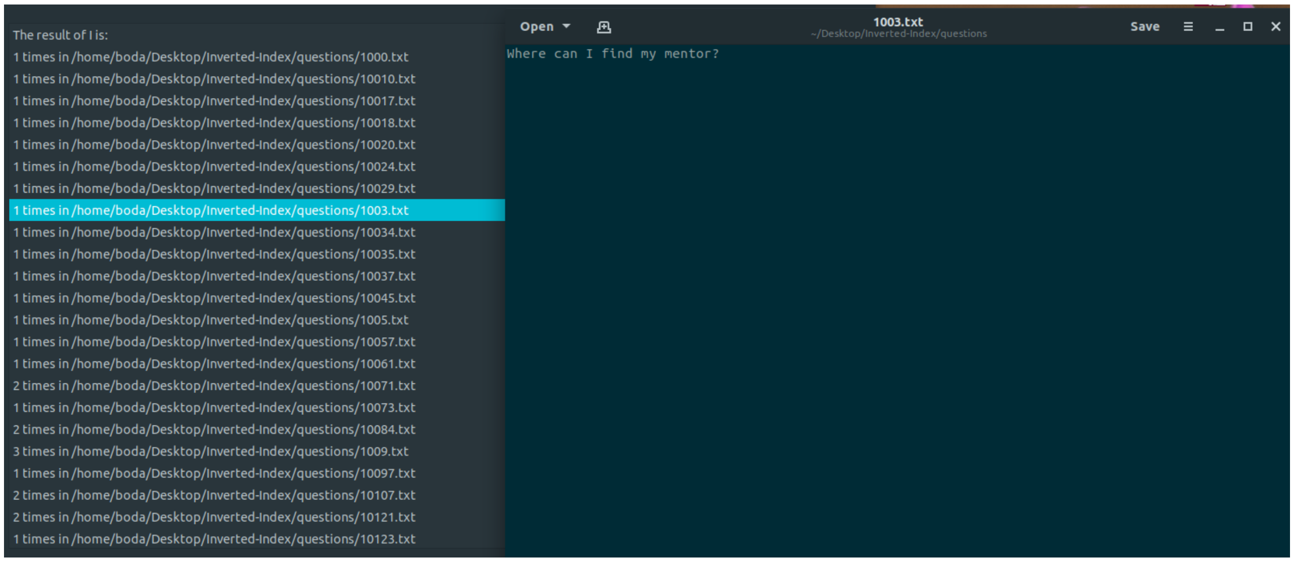
```
There's no results for ABCD
```

If there's results for such a word, it would display them in a list view:



```
The result of I is:
1 times in /home/boda/Desktop/Inverted-Index/questions/1000.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10010.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10017.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10018.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10020.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10024.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10029.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/1003.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10034.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10035.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10037.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10045.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/1005.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10057.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10061.txt
2 times in /home/boda/Desktop/Inverted-Index/questions/10071.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10073.txt
```

The user can double-click on any search result to open the file :

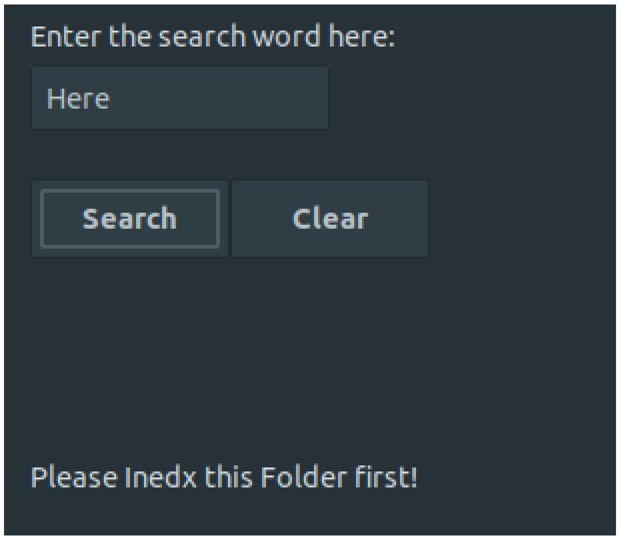


```
The result of I is:
1 times in /home/boda/Desktop/Inverted-Index/questions/1000.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10010.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10017.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10018.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10020.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10024.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10029.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/1003.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10034.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10035.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10037.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10045.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/1005.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10057.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10061.txt
2 times in /home/boda/Desktop/Inverted-Index/questions/10071.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10073.txt
2 times in /home/boda/Desktop/Inverted-Index/questions/10084.txt
3 times in /home/boda/Desktop/Inverted-Index/questions/1009.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10097.txt
2 times in /home/boda/Desktop/Inverted-Index/questions/10107.txt
2 times in /home/boda/Desktop/Inverted-Index/questions/10121.txt
1 times in /home/boda/Desktop/Inverted-Index/questions/10123.txt
```

```
Where can I find my mentor?
```

New search results would be stored on bottom of the list, so the list can be big and messy. The user can maintain it by clicking the “Clear” button which clear the results list.

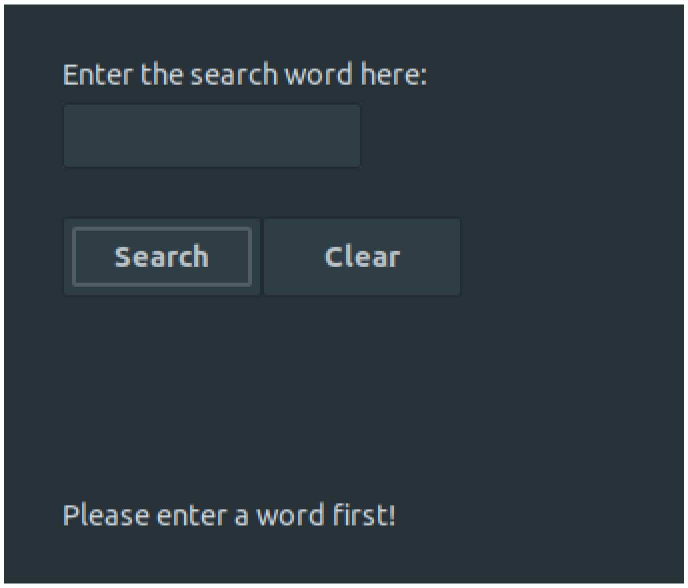
If the user didn’t index the folder and tried to search for a word, the program would tell him to index this folder first



Enter the search word here:

Please Inedx this Folder first!

If the user didn't enter any word and tried to search for empty text, the program would tell him to enter a word first:



Please enter a word first!

03

Third Topic

Implementation:

The project consists on three main parts :	**- Trie class**
	- Main class
	- App class

App Class:

This part is required for wxWidgets to know the entry point of the program, and to initialize the main frame.

It consists of two files : cApp.h, cApp.cpp

Trie Class:

This class is for the Trie. It has some data members :

- A string to int map, where we store the statistics for each word at the end node of each word.

- A char to Trie map, where we store the children for each node of the Tire.

- Bool variable to mark if the node is the end of a word.

This class have three functions:

add_word :

This function takes in the word, and the path of it's file, and add it to the Trie

Complexity : O(N), where N is the length of the word

get_word_stats :

This function takes in the words, and return a string to int map , which contains the

the statistics for this word

Complexity : O(N), where N is the length of the word.

store_trie :

This function takes in the file path, Json object, and the word.

It would go over the nodes in depth first manner, and when it reaches an end of a word,

it would add it to the Json object, and directly store the word statistics to the txt file database

By the end of it's execution, we would have two files, a Json file and a txt file.

We use the Json file as the primary database, where we store the indexing of a folder, and then read from it later instead of re-indexing the same folder again.

The txt file is like a general readable database, where we can look at the statistics for every word.

Complexity : O(N). where N is the number of nodes in the Trie.

Main Class :

This is the core of the project, where we create the GUI part, along with call the Trie functions.

Data variables :

It has data variables related to the GUI part, like a wxButton, for Buttons, ….

Functions :

is_indexed :

We call this function in many places. Its core use it to search the Database folder

And see if we already indexed the current folder.

It returns a Boolean variable, telling if this folder was indexed before or not.

on_clear_button_clicked :

This is easy function, only used to clear the results ListBox.

on_item_clicked :

This function is called when we double click any of the search results.

It would then run a bash command, to open the file we clicked at, in the default text editor.

on_browse_clicked :

This called when we click on the browse button. It opens the OS file explorer and when we chose a folder, it will write it's name in the path edit text box.

on_index_button_clicked :

This function is called when we click on the index button. It basically open the input folder, and loop over the files inside it, and then opens each one of them. Then it loop over the file and add its words in a vector, then it closes the file and add all word to the Trie using the Trie::add_word().

Then, it creates the Json and txt files and calls Trie::store_trie() function

to store the indexing to the database.

on_search_button_clicked :

This function is called when we press on the search button. It basically reads the word we entered in the edit text box. If the edit text box is empty, then it does

nothing, and notify the user to enter a word.

If the user entered a correct word, then it checks if we indexed the folder. It

goes to the Database folder and checks if we have entry with same name as the

input folder. If we successfully indexed this folder, then we search our current Trie for

the word, if we indexed this folder in earlier time, then the current Trie would be

empty, and then we would go and read from the database Json file. Then we display

the results if there's any, if not, we notify the user that there's no result for such word

read_from_json :

This functions takes in a file path, and returns a string to vector of pair of string, and int map.

So we would have for each word , a vector of pairs, where each entry in it is the file's path, and how many times this word occurred in that file.

getexepath :

This function return the path of the binary file. We use it mainly to get the program path, as we create the Database folder in the same directory as the binary file

Also it helps us, in making all files paths relative, so the project works anywhere, regardless where the binary file at.

Constructor :

where we simply initialize the GUI parts, and set the click listeners to the buttons.

04

Fourth Topic

Complexity of operation:

Complexity of Adding word to Trie :	O(N), where N is the length of the word.
Complexity of Storing the Trie to database :	O(N), where N is the number of nodes in the Trie.
Complexity of Search for Word from the Trie:	O(N), where N is the length of the word.
Complexity of Search for Word from the database :	O(log(N)), where N is the total number of words in the database.
Comment: Indexing using Trie, is a much more efficient than the naive algorithm, rather than using BST.	

Real time statistic, for the Program over the data set :

Indexing the dataset :	12 secs
Search for Word from the Trie :	0.5 sec
Search for Word from the database :	9 secs

05

Fifth Topic

Each team member role:

Abldelrahman Atef :	- Contribute to make code of Main partition and debug it. - Contribute to make Trie (the digital tree)
Omar Ahmed :	- Contribute to make code of Main partition and debug it. - Contribute to make Trie (the digital tree)
Shimaa Mohamed :	- Contribute to make code of Main partition and debug it. - Contribute to make Trie (the digital tree)
Omar Hatem :	- Contribute to make GUI - Contribute to read and generate Json data
Doaa Hesham :	- Contribute to make GUI - Contribute to read and generate Json data

06

Sixth Topic

Resources and references:

- Forward vs Inverted Indexing
- Xapian
- Trie
- Json.hpp
- wxWidgets