Please go to the last section of this file and follow the instructions under section “instructions for running this project” to run this project.

1.Description of the project: Developing Academic Search Engine using C++ and Qt.

This project intends to create a search engine that will search for resources in USC like publications, research papers, journals, conference proceedings, etc.  We want to use the qt framework to create a simple UI over the C++ application.

2.Features: The developed system supports the following features:

* This system supports full title search, partial title search, and keyword-based search, with no restriction on improper spacing or incorrect case of alphabets
* Partial title with the relative order of words maintained returns accurate results at the top of the list of results
* Keyword has to be spelled properly in the title search, other words may have some wrong spellings with the relative order of words maintained
* Returns results in the form of URL and clicking on this URL can open the research article
* To display search results the user has to type at least one keyword correctly in the search bar
* For partial title or multiple keyword-based searches from a user, the closest results displayed earlier (higher in the page) in the list of results displayed

3.Overview of implementation: The system overview is shown in fig .1. In the frontend user types the search title and presses the find button and gets the results in the form of URL. The search title input from the user is processed in the backend to get the close results and then these closed results are returned from backend to frontend to be shown to the user.

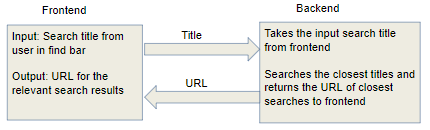


Fig.1: System overview

The searching algorithm in the backend uses a map where each entry in the map is a keyword mapped to the root of a binary search tree, the overview of the data structure is shown in fig. 2.

**map**

**“apple”**

**“analytics”**

**Keywords**

**“iot”**

**“fpga”**

**Binary Search Tree with titles containing keyword “analytics”**

**Binary Search Tree with titles containing keyword “iot”**

**Binary Search Tree with titles containing keyword “apple”**

**Root of binary search tree**

**Root of binary search tree**

**Root of binary search tree**

**Root of Binary Search Tree**

**Binary Search Tree with titles containing keyword “fpga”**

Fig.2: Overview of the data structure

The title to be searched is taken from the search bar and conveyed to the backend on the click of the find button. The dataset consists of pairs of entries where each entry: Title and URL. An example entry of the dataset is shown below:

Apple phytochemicals and their healthbenefits><https://link.springer.com/article/10.1186/1475-2891-3-5>

Searching of titles in the backend is done by using a map of (keyword, the root of binary search tree) in two stages namely, keyword search and tree access.

**First stage**: Keyword is detected from the input search of user

**Input search**:

Enzymatic browning reactions in apple and apple products

**Keyword match**:

Some keywords have been selectively stored in the program, each keyword forms a group of titles that contain the keyword, the titles in that keyword group stored in the same binary search tree.

**Second stage**: After finding keyword in the title, access nodes of the tree for that keyword to search for the title.

**Input search**:

Enzymatic browning reactions in apple and apple products

Keyword match from stage 1

**Tree access**: map(“apple”, Root of BST for titles with “apple”)

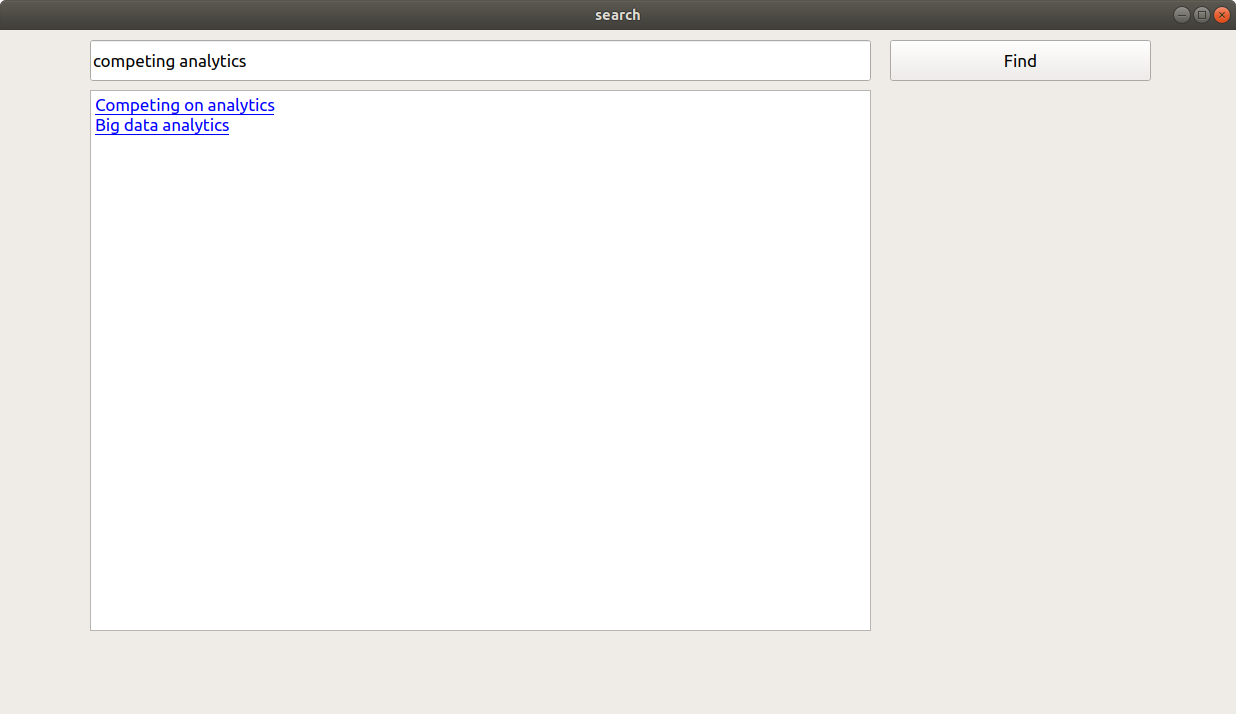
BST for “apple” stores titles containing the keyword “apple”, one BST for all titles with the same keyword, similar BSTs formed to group all titles with the same keyword.

4.Link to the video: Please follow the link below for a demonstration on the working mechanism:

<https://drive.google.com/open?id=1Uo7F7tYOvzvR2rNhliaYocJNr7lWBjb7>

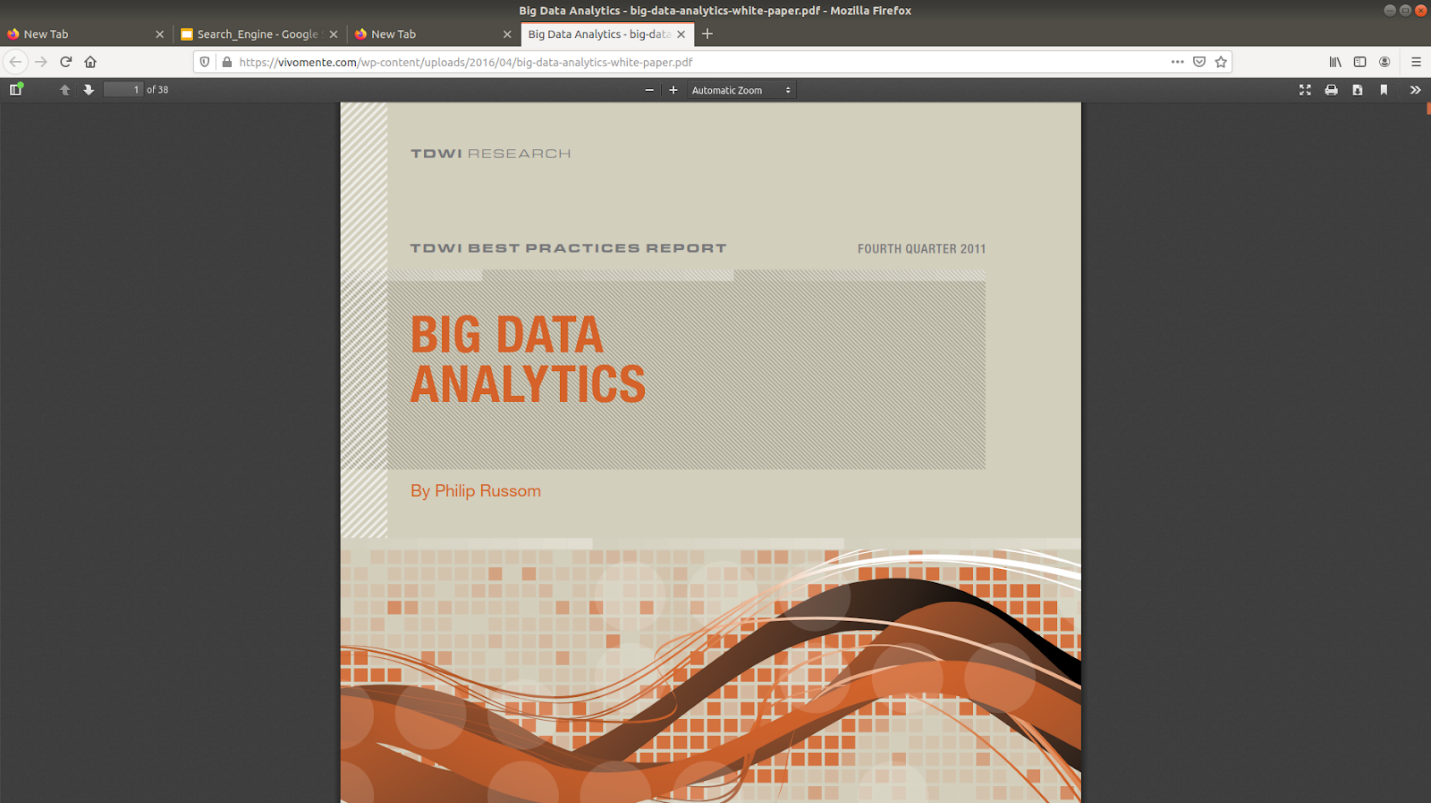
Link to presentation slides: <https://drive.google.com/open?id=13bIBAWl7AU8YjSj4_F4oLSeW2YoJDWWYufaamMbe1gQ>

5.Screenshots of the demo: This is a screenshot of the frontend Qt window in fig. 3, at the top is the search bar where a user enters the search title and clicks on the Find button in the right, then the results are displayed as URL in the bigger box below.



Fig, 3: Front window for search engine

The following page in fig. 4 appears when the user clicks on the second (from top) URL of the above search results.

Fig. 4: This page opens after the user clicks on the link second from top

6. Future work: The following points are the possible extensions to the future scope of this project:

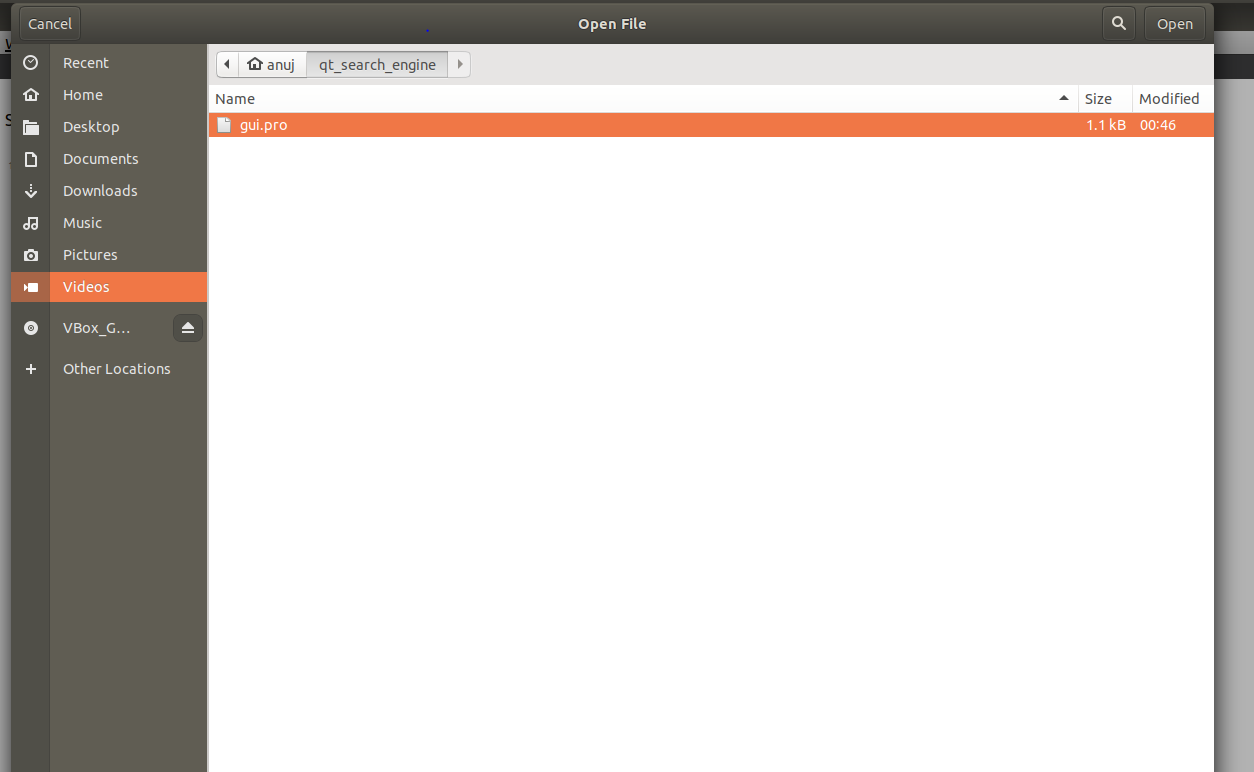
* The current system has been tested on a small dataset with a small number of keywords chosen, further with a larger dataset keyword selection can be made adaptive
* Future improvement in the search mechanism to show accurate results for partial title search with no restrictions in the order of typed words
* Predictions for full titles when the user is typing the title in the search bar and these predictions can be shown as autocomplete results
* Handling the spelling errors by showing close search title (with corrected spelling) result that might be the desired search from the user

7. Instructions to run this project: Please follow the below steps for installing qtcreator:

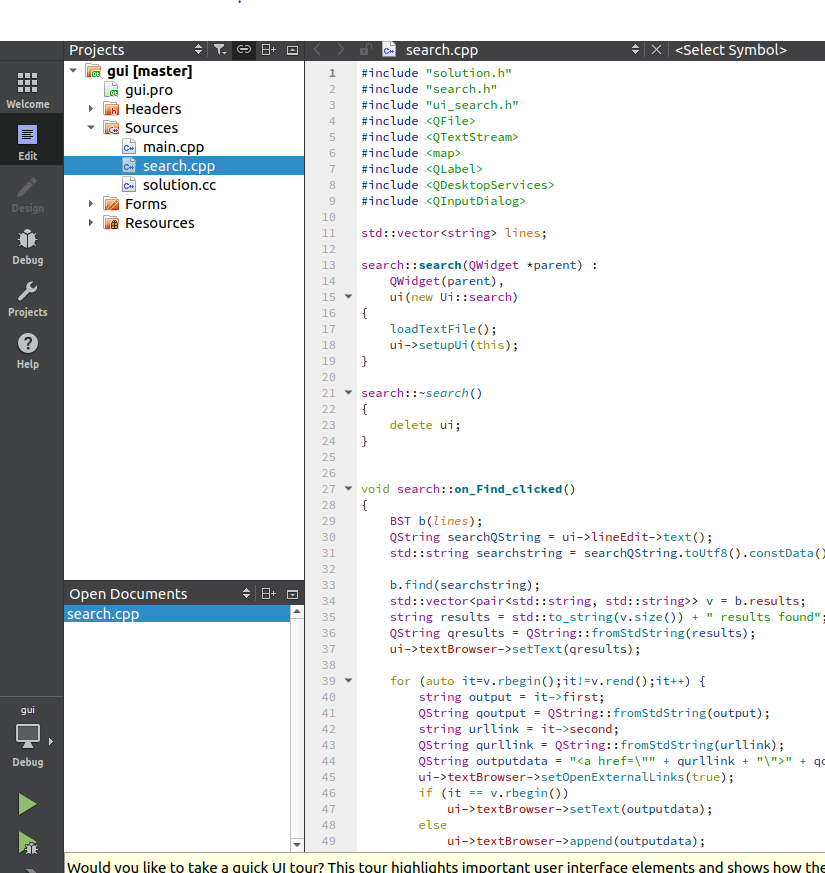
1. wget http://download.qt.io/official\_releases/qt/5.14/5.14.2/qt-opensource-linux-x64-5.14.2.run
2. chmod +x qt-opensource-linux-x64-5.14.2.run
3. ./qt-opensource-linux-x64-5.14.2.run
4. Please follow the below video for instructions on installing qtcreator: <https://youtu.be/I5jasWrsxT0>
5. sudo apt-get install build-essential
6. sudo apt-get install libfontconfig1
7. sudo apt-get install mesa-common-dev
8. sudo apt-get install libglu1-mesa-dev -y

Follow the below steps for running the project:

1. Go to File, then click on Open file or project.
2. Then open the gui.pro file from the cloned project.



1. Click on the Run button as shown below



**Click this button**