“Google @ IITB”

Akash S. Doshi 140010008
Ch. Jyothi Durga Prasad140010042
Aditya Bhosale 140010009
Mainak Majumdar 14D110022

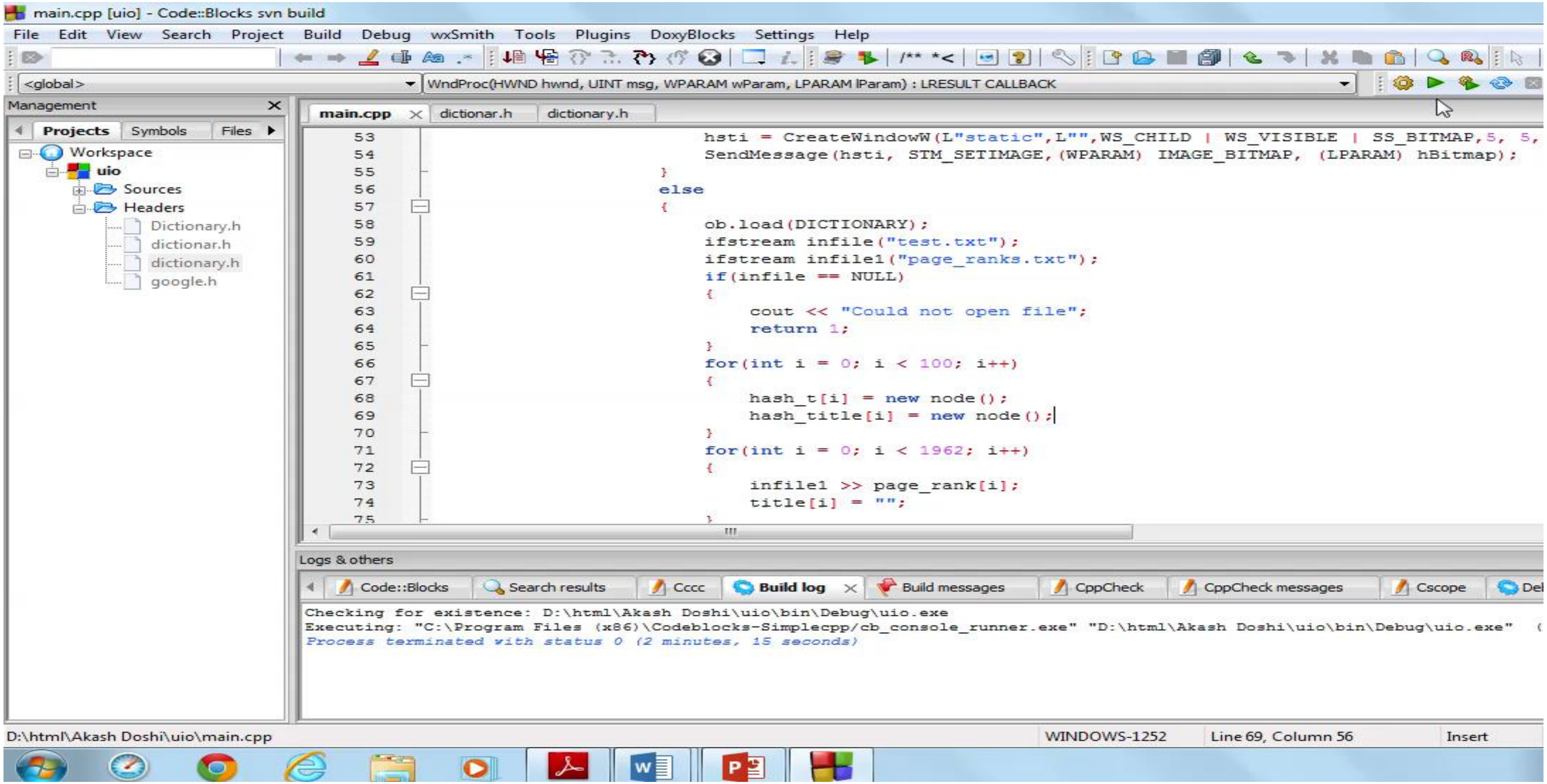
Problem Statement & Algorithm:

- We are designing a search engine to closely model Google and its page rank algorithm.
- Using a web crawler we are obtaining all the hypertext references and their titles from the IITB main webpage, convert it into a text file and read that text file in a c++ program.
- The keywords and their “n-grams” , obtained from modification of the url’s, will be stored in the form of an array of linked lists - “hash table”.
- When the user enters a search ,the search will be n-grammed and each n-gram’s hash value will be extracted and the word will be searched for in the corresponding index of the array.
- On success the user will be shown a list of urls stored under the keyword in decreasing order of authoritativeness. In case the keyword is not found, the user will be redirected to a google search which jQueries the string entered by the user concatenated with ‘iitb’.

Problem Statement & Algorithm:

- The search entered by the user is also subjected to a spell check.
- The dictionary for the spell checker is also a text file which is stored in the form of a hash table. We have implemented a spell checker for up to one spelling mistake per word . Starting from the last alphabet, each letter will be replaced by all 26 options and the first correct word will be returned to the user, who then has the option of either searching for the spell checked word or what he entered
- The graphics for this search engine incorporate Win32 GUI, which is akin to Windows 2000.
- We have created buttons , titles and text box fields , to provide Previous and Next Page functionality, accept the search, input of the spell checker and hypertext references to the websites . One can also clear his search and write the new search without having to close the application.
- For the above, we have made use mainly of the fact that in win32 GUI, every form of input , be it from the keyboard or the mouse results in a message being sent to the CALLBACK function, and by appropriate define statements in the pre-processor , we can adjust these values to check for them correctly when received.

Video of Functional Search Engine.



```
        hsti = CreateWindowW(L"static",L"",WS_CHILD | WS_VISIBLE | SS_BITMAP,5, 5,
        SendMessage(hsti, STM_SETIMAGE,(WPARAM) IMAGE_BITMAP, (LPARAM) hBitmap);
    }
    else
    {
        ob.load(DICTIONARY);
        ifstream infile("test.txt");
        ifstream infile1("page_ranks.txt");
        if(infile == NULL)
        {
            cout << "Could not open file";
            return 1;
        }
        for(int i = 0; i < 100; i++)
        {
            hash_t[i] = new node();
            hash_title[i] = new node();
        }
        for(int i = 0; i < 1962; i++)
        {
            infile1 >> page_rank[i];
            title[i] = "";
        }
```

```
Checking for existence: D:\html\Akash Doshi\uio\bin\Debug\uio.exe
Executing: "C:\Program Files (x86)\Codeblocks-Simplecpp/cb_console_runner.exe" "D:\html\Akash Doshi\uio\bin\Debug\uio.exe"  (
Process terminated with status 0 (2 minutes, 15 seconds)
```

Challenges:

- Google page rank algorithm ranks pages depending on how many pages give link to that page and also takes into consideration the authoritativeness of the pages that give the link.
- However, there’s a problem of dangling nodes (also called sink) ie. Pages which have only incoming links without any going out. This problem is solved by transforming the original matrix into what is called the Markov matrix which replaces the columns whose all entries are zero by 1/n (n is the number of pages in the database)
- Another problem is that of disconnected graph, meaning an isolated set of links in the database. In this case, the eigenvalue equation gives ambiguous result (more than one eigenvector for one eigenvalue). To solve this problem, we take into consideration a damping factor (0.85 in our case)
- $G = (1-d)M + dS$ G is called the Google matrix

  M is the Markov matrix

  d is the damping factor

  n is the total number of pages in the database

  S is an nxn matrix with all entries = 1/n

Future Work & Innovation:

- Currently on searching for “computer science” no CSE website is returned as neither the website nor the title contain computer rather they contain “CSE”. If we were also able to hash the text displayed on the webpage this problem could be solved, but implementation of “Big Data” structures on this scale is not feasible.
- The implemented page rank algorithm depends only on the number of hypertext references on a page . By obtaining other data, like number of people having visited a page, standard of web designing incorporated in the page, a more powerful page rank algorithm can be obtained akin to the ones used today by popular search engines.

Future Work & Innovation:

- As soon as the user enters about three alphabets, a drop down menu can be displayed (using the CBS_DROPDOWNLIST function in Win32) to show him the possible search words . This can be done as our hash table also stores the n-grams of every word and hence the word itself can be traced.
- To provide for reusability of this code, an algorithm has been developed to provide the contents of the websites and the dictionary in a read-only mode to a future user . This user will be provided with an authorization code for a Dropbox account which contains these files . On entering the code, the program will search that Dropbox account for the requisite files, and read them into the program, without showing the user it’s contents.