**CSC326**

**Lab 2 Report**

**Development Phase 2**

Group #: 4

Names: Anny Kim Ly (997581718)

Kevin Gumba (997585117)

**Section 1: Frontend (by Anny Ly)**

1.0 Design Decisions

*1.0.1 Results Table*

The front end accesses the database created by the back end and returns a list of URLs that contains the first word of the query string. The URLs,ordered by their pagerank, are then displayed in a table format along with their title and description (BONUS).

The results table is vertically scrollable for when the 10 results are too big to fit on 1 page. The length varies depending on the descriptions of the URLs.

Pagination was used to limit the number of results displayed. A maximum of 10 URLs can be displayed per page, with the last page containing the remaining results. At the bottom of the search page, page buttons are displayed to navigate through the results.

*1.0.2 Query Interface*

On the results page, a query interface is displayed at the top of the webpage for easy searching of new keywords.

*1.0.3 Error Page*

If the user tries to access a web page or a file that does not exist on the website, they will be redirected to the error page that includes a button that can return them to the homepage.

1.1 Instructions to Run Code

**Note:** Please run “python crawler.py” in the Backend folder first before running SpaceExplorer.py. A dbFile.db must first be created in order to access persistent data.

To run the program, **simply type “python <file name>.py” in the command line** to start up the search webpage. In this case the file name is “SpaceExplorer.py”. On execution of the file, the query page can be accessed through “<http://localhost:8080/>”. The user will be presented with a simple interface to submit a keyword or phrase. Once the “search” button is clicked, the user will be redirected to the results page. There, they can then find a results table with a list of URLs ordered by their page rank along with additional detail such as title and description (BONUS). If there are more than 10 results, then pagination will occur. The first 10 results will be listed on page 1 and the next 10 results on page, etc. A maximum of 10 results are displayed per page, with the last page containing the remaining results (which will be less than or equal to 10).

**Section 2: Backend (by Kevin Gumba)**

2.0 Design Decisions

*2.0.1 Pagerank Algorithm*

An iterative damping factor is used in order to calculate the page ranks of the urls. The calculation is shown below from <http://en.wikipedia.org/wiki/PageRank> under Computation, Iterative:  
 

PR is the page rank of a given url, d is the damping factor, and N is the amount of outbound links for that page. A summation of the inbound links’ page ranks divided by the amount of outbound link they have are included.  
  
In order to determine the pagerank for all urls found within the depth set for the crawler, a new dictionary data structure is used.

* outboundLinks
  + contains the outbound links for a given url
  + one dictionary data structure
    - key: doc id
    - values: corresponding outbound links’ doc ids

The pageranks are calculated once the crawler and parsing are complete. The function rankPage is called. rankPage first calculates the inbound links in a dictionary data structure, calculates a preliminary pagerank for all urls, with PR = (1 - d) / N with d = 0.85. After, the whole pagerank algorithm is done 20 times.

The drawback for this algorithm is that it is not memory efficient since three data dictionaries are created. However, the code allows for easy readability and debugging.

*2.0.2 SQLite3 Persistent Storage*

All dictionary data structures are stored using sqlite3. All dictionary data structures are stored in their own table. This allows for simplicity of code for easy debugging and data handling in the front end. The drawback is the inefficiency use of memory since many tables are created.

Storing of data structures with SQL is done once the crawler and parsing is complete. Function storeInSQL is called. Tables are first created if they do not currently exists. This allows to preserve and not overwrite old tables. Later, checks using insert or update are used in order to determine if a tuple exists or not.

*2.1 Instructions to Run Code*

To run the function, **simply type “python <file name>.py” in the command line** to receive the output.

* crawler.get\_page\_ranks()
  + lab2tester1.py
  + Returns a dictionary with page ranks. Print page ranks for sites found within depth 1.
* crawler.get\_SQL\_from\_word(word)
  + lab2tester2.py
  + Outputs all information stored in SQL corresponding to word ”about”. Prints corresponding url, title, description, and page rank