This Search Engine has 5 parts. They are:

1) Crawler,

2) Database (handled using files),

3) Inverted Index,

4) Server,

5) Client End (handled using both JavaFX application and a Tomcat server).

- ***Crawler:***

1) The crawler that takes an URL as a seed.

2) Then it connects to the URL using http connection.

3) Retrieves the html file of the page.

4) Parses the html file and collects more available URLs in that page.

5) Stores the text of that page in the database (here, it is done using "text files").

6) Then the crawler connects to the collected URLs one by one and repeats the above steps again and again until there are no more URLs to visit. The repetition of same URL is stops effectively.

7) As it is a multi-threaded application, several crawling processes can be run parallel to each other.

- ***Database (handled using files):***

1) The texts, saved by the crawler from the visited web pages, are stored here for further process. The each file stores the URL and the texts from that web page.

- *Inverted Index:*

  1) An individual program does the job of indexing.

  2) The text files which are saved by the crawler, are accessed simultaneously one after another.

  3) A Hashtable is used for indexing. In the table, for each value another Hashtable is used so that here the URLs can be stored with the word-frequency.

  3) In a file, each word is indexed in such a way that, the key of the index is the word and the value is the table with URLs and word-frequencies.

- *Server:*

  1) This server deserializes the inverted index.

  2) Works as a server to receive clients' request (the phrase to search), searches and sorts the results from the most relevant to least relevant.

  3) Sends the results as an array-list to the client end.

- *Client end:*

  1) The user inputs a phrase in a text field, and this program sends it to the server.

  2) Receives the result and shows it in a new webpage in the default browser.

  3) User can enter search-word in a webpage (which is linked up with a servlet and hosted by a tomcat server) and also in a JavaFX application.

Problems I had to face while working with this project and how I solved them:

1) To crawl Bangla sites I had to change the regex.

2) For indexing at first I used word positions, but to reduce the memory consumption, I used word diagram latter.

3) To run multiple crawling and indexing parallel to each other, I used multiple threads instead of one.

4) In order to avoid heap memory exceeding, I manually increased the heap size of the programs.

5) The larger the index is, the more time and memory it consumes while being deserialized. So solve the issue I manually increased the heap memory to run the server.

Possible developments:

1) Auto-correction of words can be implemented.

2) Loading the index can be made faster in future.

3) The sorting algorithm can be developed.

4) More specific results can be show at the very top of the result list.

5) For a certain word, the hit-count can be considered. While a certain URL gets more hits than the others, it should get a higher priority. The next time its position will be higher in the list.

6) While retrieving words, words with similarities can be identified (example: read, reading).

7) The files can be replaced by a Database.

Java Commands:

- To compile all the file:
  - javac *.java
  - javac clientforsearchengine*.java
- To run Crawler Bot:
  - java URL_checker
- To run Inverted-Indexing program:
  - java MainI
- To run Server:
  - java MainOfSorting
- To run Client-End program (JavaFX application):
  - java clientforsearchengine.ClientForSearchEngine
- To run Client-End program (In a browser as a web page ("servlet" used)):
  - Run the tomcat server
  - Go to: "http://localhost:8080/Tomcat(test5)/SearchPage.jsp"