Index.html: This is main project screen while start the project and contains link to add new URLs and Search for keywords.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Search Engine</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<a href="index2.html">Add new URL</a>
<h1>Search Engine</h1>
        <form action="WebSearch" method="GET">
    <h2> <input type="text" name="search_term" class="search"/>
    <input type="image" value="submit" src="image/search_image.jpeg" height="50px"/></h2><br>
                </form>


</body>
</html>
```

index2.html: This is screen to add new URLs and contains link to go back to search.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Add new URL</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<a href="index.html">Go to Search</a>
<h1>Add new URL</h1>
            <form action="WebCrawler" method="GET">
      <h2> <input type="text" name="new_url" class="search" width= "700px"/>
      <input type="image" value="submit" src="image/add_url.png" height="50px"/></h2><br>
                        </form>


</body>
</html>
```

WebCrawler.java: This is Java Servlet, project will redirect here while adding new URLs from index2.html, this Servlet passes the entered URL to InformationCollection java class for crawling.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class WebCrawler
 */
@WebServlet("/WebCrawler")
public class WebCrawler extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * Default constructor.
    */
   public WebCrawler() {
      // TODO Auto-generated constructor stub
   }

	/**
	* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	*/
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub
		response.setContentType("text/html");

		PrintWriter out = response.getWriter();
		String title = "Add new URL";
		String url = request.getParameter("new_url");

		String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
		 "transitional//en\">\n";

		out.println(docType + "<html>\n" +
		 "<head><title>" + title + "</title><link rel=\"stylesheet\" href=\"style.css\"></head>\n"
+
		 "<body><h2 align=\"center\" display=\" block\">" + "URL Added: " +url+ "</h2>\n" +
		 "<h2><a href=\"index2.html\" align=\"center\" >Add more URL</a></h2><br><br>\n"+
		 "<h2><a href=\"index.html\" align=\"center\">Go to Search</a></h2>\n"+
		 "</body></html>");

		InformationCollection Info = new InformationCollection();
			Info.search(url,request.getHeader("User-Agent"));


	}

	/**
```

```
	* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	*/
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub
		doGet(request, response);
	}

}
```

WebSearch.java: This is Java Servlet, project will redirect here while search for keywords from index.html, this Servlet passes the entered keyword to ResponseHandle get in returns all the URLs containing that keyword to WebSearch Servlet. Then servlet will print out all the URLs as in link format so that by clicking on them we can redirect to that page.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Iterator;
import java.util.Set;
/**
 * Servlet implementation class WebSearch
 */
@WebServlet("/WebSearch")
public class WebSearch extends HttpServlet {
	private static final long serialVersionUID = 1L;

  /**
   * @see HttpServlet#HttpServlet()
   */
  public WebSearch() {
    super();
    // TODO Auto-generated constructor stub
  }

	/**
	* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	*/
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub
		response.setContentType("text/html");

		PrintWriter out = response.getWriter();
		String title = "Search Engine";

		String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
		 "transitional//en\">\n";

		out.println(docType + "<html>\n" +
		 "<head><title>" + title + "</title><link rel=\"stylesheet\" href=\"style.css\"></head>\n"
+
		 "<body><a href=\"index.html\">Search Again</a><br><h1 align=\"center\">" +
		 "Search Result for: " +request.getParameter("search_term")+ "</h1>\n");

		String search_string = "";

			search_string = request.getParameter("search_term");
			Set<String> urls = ResponseHandle.search(search_string);
			Iterator<String> iterator = urls.iterator();
			if(!iterator.hasNext()) {
```

```
                                        out.println("<br><br><h1>No Results found</h1>");
                                }
                                while (iterator.hasNext()) {
                                        String nextURL = iterator.next();
                                        out.println("<h4><a href=\""+nextURL
+"\">"+nextURL+"</a></h4>");

                                }

                out.println("</body></html>");
        }

        /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                // TODO Auto-generated method stub
                doGet(request, response);
        }

}
```

InformationCollection.java: This class has defined limits that It crawls maximum 5 pages and collect information by passing URL to ResponseHandle class in such a way that It checks for any page is already visited or not and If not visited then visit that page and find a word, otherwise go to next page.

```
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;

public class InformationCollection
	{
	private static final int MAX = 5;
	private List<String> pagesToVisit = new LinkedList<>();
	private Set<String> visitedPage = new HashSet<String>();


	public void search(String url,String user_agent)
		{
		while(this.visitedPage.size()< MAX)
			{
			String currentUrl;
			ResponseHandle Response = new ResponseHandle();

			if(this.pagesToVisit.isEmpty())
				{
				currentUrl = url;
				this.visitedPage.add(url);
				}
			else{
				currentUrl = this.nextUrl();
				}

			Response.crawl(currentUrl,user_agent);

			this.pagesToVisit.addAll(Response.getLinks());
			}
		}

	private String nextUrl()
		{
		String nextURL;
		do
			{
				nextURL = this.pagesToVisit.remove(0);
			}while(this.visitedPage.contains(nextURL));

		this.visitedPage.add(nextURL);

		return nextURL;
		}

}
```

ResponseHandle.java: This class handles http request and solve them using jsoup.jar external archive jar file. We can get html page from this in String and then we are going to process that page using compressed trie. This class also search for keyword from the compressed trie and return back the URLs containing entered keyword.

```
import java.io.IOException;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class ResponseHandle
	{
	private List<String> links = new LinkedList<String>();

		public void crawl(String url,String user_agent)
		{
		try {
			Connection connection = Jsoup.connect(url).userAgent(user_agent);
			Document HTMLDocument = connection.get();

			String page = HTMLDocument.text();

			URLProcess p = new URLProcess();
			p.addPagewithURL(page, url);

			Elements linkOnPage = HTMLDocument.select("a[href]");

			for(Element link : linkOnPage)
			{
				this.links.add(link.absUrl("href"));
			}

		} catch (IOException ioe) {

			System.out.println("HTTP request not successful " + ioe);
		}
	}

	public static Set<String> search(String line) {
		String[] words = line.split(" ");
		return searchWords(words);
	}

	public static Set<String> searchWords(String[] words) {
		Set<String>result = new HashSet<>();
		for (String word : words) {
			Set<String> wordResult = searchWord(word);
			if (wordResult != null) result.addAll(wordResult);
		}
		return result;
```

```
	}

	public static Set<String> searchWord(String searchWord){

		char[] words = searchWord.toCharArray();
		CompressedTrie root = URLProcess.trie;
		for (int i = 0; i < words.length; i++) {
			root = root.getChild(words[i]);
			if (root == null) return null;
		}
		return root.urls;

	}

	public List<String> getLinks(){
		return this.links;
	}

}
```

URLProcess.java: In this class, we are processing compressed trie b adding all the words from web page of the URL and URL itself and comparing and filtering of stopping words which are excluded from the page which is collected for finding searching words.

```
public class URLProcess {
	public static CompressedTrie trie = new CompressedTrie('$');

	public void addPagewithURL(String page, String url) {

		String[] words = page.split(" ");
		for(String word : words){
			if(word.trim().length()>0  && WordFilter(word) &&
!this.isStoppingWord(word)){
				WordProcessing(word.trim(), url);
			}
		}
	}

	private void WordProcessing(String w, String url) {
		String word = w.toLowerCase();
		int i = 0;
		CompressedTrie t = trie;
		while(i < word.length()) {
			t.addChild(word.charAt(i));
			t = t.getChild(word.charAt(i));
			i++;
		}
		t.addUrls(url);
	}

	private boolean WordFilter(String word) {
		if( word.equals("|")|| word.equals("+")|| word.equals("-")
				|| word.equals("#")|| word.equals("$")|| word.equals("%")
				|| word.equals("^")|| word.equals("&")|| word.equals("*")
				|| word.equals("(")|| word.equals(")")|| word.equals("_")
				|| word.equals(",") || word.equals(".") || word.equals("/")
				|| word.equals("==") ||word.equals("//") || word.equals("@")
				|| word.equals("\"")|| word.equals("[")|| word.equals("]")
				|| word.equals("\\")|| word.equals("/{")|| word.equals("}")
				|| word.equals("<")|| word.equals(">")|| word.equals("?")
				|| word.equals(";")|| word.equals("'")|| word.equals(":")
				|| word.equals("=")|| word.equals("!")
				){

			return false;
		}
		return true;
	}


	public static String[] alphabet = {
			"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k",
			"l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v",
			"w", "x", "y", "z"
```

```
	};

	public static String[] pronouns = {"i","we","they","her",
				"him","them","his","their","you","he","she","it",};

	public static String[] articles = {"the", "a", "an"};


	public static String[] prepositions = {

	"concerning","considering","off","aboard","about","above","across","after",
				"near", "of","on", "onto", "over",
"opposite","outside","past","per","plus","save",
				"beside","besides","between","beyond","but","by","despite", "down",
"during",
				"except", "following","for", "from","like", "minus","in", "inside","into",

	"against","along","among","around","as","exclude","at","before","behind","below",
				"up", "upon", "via","versus","with", "within", "without","unlike", "until",
				"since", "than", "to","through", "toward", "towards", "under"

	};


	public static String[] html = {
				"a","abbr","b","base","button","cite","code","class", "href", "p",
"head","div",
				"html", "http", "body","figure","span",  "id", "name", "target", "style",
"img", "src",
				"border","li", "width", "height","menu","q","small","align"
	};


	public boolean isStoppingWord(String w) {
		String word = w.toLowerCase();
		for (String str : articles) {
			if (word.equals(str)) return true;
		}
		for (String str : pronouns) {
			if (word.equals(str)) return true;
		}
		for (String str : prepositions) {
			if (word.equals(str)) return true;
		}
		for (String str : html) {
			if (word.equals(str)) return true;
		}
		for (String str : alphabet) {
			if (word.equals(str)) return true;
		}
		return false;
	}

}
```

CompressedTrie.java: In this class, we are implementing compressed trie for all words and URLs and arrange them in specific order for easy searching.

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class CompressedTrie {
	public List<CompressedTrie> children;
	public char val;
	public Set<String> urls;
	public CompressedTrie(char v) {
		children = null;
		urls = null;
		this.val = v;
	}
	public void addUrls(String url) {
		if (urls == null) urls = new HashSet<>();
		if (!urls.contains(url)) urls.add(url);
	}

	public Set<String> getUrls() {
		return urls;
	}

	public void addChild(CompressedTrie trie) {
		if (children == null) {
			children = new ArrayList<>();
		}
		if (!hasChild(trie)) {
			children.add(trie);
		}
	}

	public void addChild(char c) {
		if (children == null) {
			children = new ArrayList<>();
		}
		if (!hasChild(c)) {
			children.add(new CompressedTrie(c));
		}
	}

	public boolean hasChildren() {
		return children != null;
	}

	public boolean hasChild(CompressedTrie trie) {
		if (children == null) return false;
		return children.contains(trie);
	}

	public boolean hasChild(char c) {
		if (children == null) return false;
		for (CompressedTrie trie : children) {
```

```
                        if (trie.val == c) {
                                return true;
                        }
                }
                return false;
        }

        public CompressedTrie getChild(char c) {
                if (children == null) return null;
                for (CompressedTrie trie : children) {
                        if (trie.val == c) {
                                return trie;
                        }
                }
                return null;
        }
}
```