**TOPICAL SEARCH ENGINE**

**HUB FOR TUTORIALS**

# PROJECT REPORT

Submitted for the course: Web Mining (CSE3024)

By

**15BCE0329 (Harshit Kedia)**

**15BCE0561 (Soumallya Boral)**

**15BCE0957 (Shashwat Mishra)**

**15BCE2013 (Satyaki Mukherjee)**

Slot: B1

## Name of Faculty: Prof. Thendral P

**(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)**



November, 2017

**INTRODUCTION TO TROPICAL CRAWLERS:**

Tropical or focused crawlers are web crawlers that collects web pages according to the relevance of it with a particular topic or web page. It has a seed URL or a topic given to with a frontier database that it can access to crawl pages. Its major work is just to find relevance among the pages or crawl through pages following a certain distance-significance factor. The URL if never been crawled before, gets into a database table called frontier.

The crawler that we have created here is named “MAKBOT/1.0” which crawls through an array of seed URL’s and find all the significant URL’s to them.

**IMPLIMENTATION:**

1. Extracting the data using PHP.
2. Extracting the links using PHP.
3. Frontier database creation.

**DATA TO BE EXTRACTED:**

1. Title of the webpage.
2. Description of the webpage.
3. Keywords given in the webpage.
4. URL’s of new webpages used in the web page (outlinks).

**CODE EXPLANATION:**

The following code is used to crawl a given web page and extract a given set of information from it. It implements a topical crawler, which collects web pages according to the relevance of the given pages and topics. The crawling begins at a seed URL. The major significance of this type of crawler is that it crawls pages in order of their relevance.

The code below defines three seed pages to start the crawling from. As we are focussing particularly on tutorial websites, we are using three tutorial sites as the three seed pages.

// This project is for web mining j component.

// this code is used to crawl the web page and extract information.

// its a tropical crawler.

// it follows breadth first algorithm.

<?php

$start = "https://www.w3schools.com/";

$start = "https://www.tutorialspoint.com/";

$start = "https://csstutorial.net/";

As we start from the seed URL and traverse other pages, we require a database to store the crawled pages, un-visited pages as well as the keywords and descriptions of the pages that have already been traversed. Any URL that has never been crawled before is put into a database called frontier. In the following code, we connect to the local host and connect the database. A query is used to insert the keywords and descriptions of pages being crawled into the databse. Separate arrays are maintained for URLs that have been crawled and URLs which are yet to be crawled.

//Databse initialisation

$dbhost="localhost";

$dbuser="root";

$dbpass="";

$dbname="web";

$connection=mysqli\_connect($dbhost,$dbuser,$dbpass,$dbname);

if(mysqli\_connect\_errno()){

die("Database connection failed:"

);

}

$query="insert into frontier(title,description,keywords,url) values('mak','mak','mak','mak.com');";

$already\_crawled = array();

$crawling = array();

While crawling URLs, the most important part is to extract the required data from them. The code below extracts the details of the URL using get\_details function. A global connection is established and the details are then extracted – title, description and keywords, all of which can be obtained from the URL using get\_attribute. The title, description, keywords and URLs are then added to the database using mysql\_real\_escape\_string.

// this function extracts the detail of the url given to it....

function get\_details($url) {

global $connection;

$options = array('http'=>array('method'=>"GET", 'headers'=>"User-Agent: MAKBOT/1.0\n"));

$context = stream\_context\_create($options);

$doc = new DOMDocument();

@$doc->loadHTML(@file\_get\_contents($url, false, $context));

//extracting title, description and keywords

$title = $doc->getElementsByTagName("title");

$title = $title->item(0)->nodeValue;

$description = "";

$keywords = "";

$metas = $doc->getElementsByTagName("meta");

for ($i = 0; $i < $metas->length; $i++) {

$meta = $metas->item($i);

if (strtolower($meta->getAttribute("name")) == "description")

$description = $meta->getAttribute("content");

if (strtolower($meta->getAttribute("name")) == "keywords")

$keywords = $meta->getAttribute("content");

}

$title = mysql\_real\_escape\_string($title);

$description = mysql\_real\_escape\_string($description);

$keywords = mysql\_real\_escape\_string($keywords);

$url = mysql\_real\_escape\_string($url);

Debugging is an essential part of any robust code. The following code provides the simplest form of debugging using echo statements simply to ensure that the correct data has been extracted from the URLs.

//for debugging...

echo "$title</br>";

echo "$description</br>";

echo "$keywords</br>";

echo "$url</br>";

The code below accesses the databases of URLs and has queries to insert the details of new URL’s into the database. It displays appropriate error messages if there are problems with accessing the database. After this, the title, description and keywords are returned.

//database accessing and insertion...

$query="INSERT INTO frontier(title,description,keywords,url) values('{$title}','{$description}','{$keywords}','{$url}');";

//$query="INSERT INTO frontier(title,description,keywords,url) values($title,$description,$keywords,$url);";

$result=mysqli\_query($connection,$query);

if($result)

$contact="success";

else

$contact="failure";

echo "$contact.<p></p>";

//returning the title, description and keywords...

return '{ "Title": "'.str\_replace("\n", "", $title).'", "Description": "'.str\_replace("\n", "", $description).'", "Keywords": "'.str\_replace("\n", "", $keywords).'", "URL": "'.$url.'"},';

}

The code below extracts links from the URL provided and prepares for further crawling. To extract the next URL it acceses the href field in the ‘a’ attribute of the given html code. After extracting the URL, it must check whether the URL is valid or not by checking it against the format of a valid URL. After that, it adds special characters before and after for the link to run directly in the browser address bar. After this, the same process is repeated for the URLs connected to the new page.

// this function extracts all the links from the url provided to it.

function follow\_links($url) {

global $already\_crawled;

global $crawling;

$options = array('http'=>array('method'=>"GET", 'headers'=>"User-Agent: MAKBOT/1.0\n"));

$context = stream\_context\_create($options);

$doc = new DOMDocument();

@$doc->loadHTML(@file\_get\_contents($url, false, $context));

$linklist = $doc->getElementsByTagName("a");

foreach ($linklist as $link) {

// saving the link or url in $l variable...

$l = $link->getAttribute("href");

// editing the link for runing it in the browser....

if (substr($l, 0, 1) == "/" && substr($l, 0, 2) != "//") {

$l = parse\_url($url)["scheme"]."://".parse\_url($url)["host"].$l;

} else if (substr($l, 0, 2) == "//") {

$l = parse\_url($url)["scheme"].":".$l;

} else if (substr($l, 0, 2) == "./") {

$l = parse\_url($url)["scheme"]."://".parse\_url($url)["host"].dirname(parse\_url($url)["path"]).substr($l, 1);

} else if (substr($l, 0, 1) == "#") {

$l = parse\_url($url)["scheme"]."://".parse\_url($url)["host"].parse\_url($url)["path"].$l;

} else if (substr($l, 0, 3) == "../") {

$l = parse\_url($url)["scheme"]."://".parse\_url($url)["host"]."/".$l;

} else if (substr($l, 0, 11) == "javascript:") {

continue;

} else if (substr($l, 0, 5) != "https" && substr($l, 0, 4) != "http") {

$l = parse\_url($url)["scheme"]."://".parse\_url($url)["host"]."/".$l;

}

The code below checks if a given URL has already been crawled. Depending on whether it has already been crawled or not, the details of the URL will be displayed. A databse will either be created on the localhost to store all the URLs which have already been crawled or it will check from the relevance ranking table.

if (!in\_array($l, $already\_crawled)) {

$already\_crawled[] = $l;

$crawling[] = $l;

echo get\_details($l)."<p></p>";

}

}

array\_shift($crawling);

foreach ($crawling as $site) {

follow\_links($site);

}

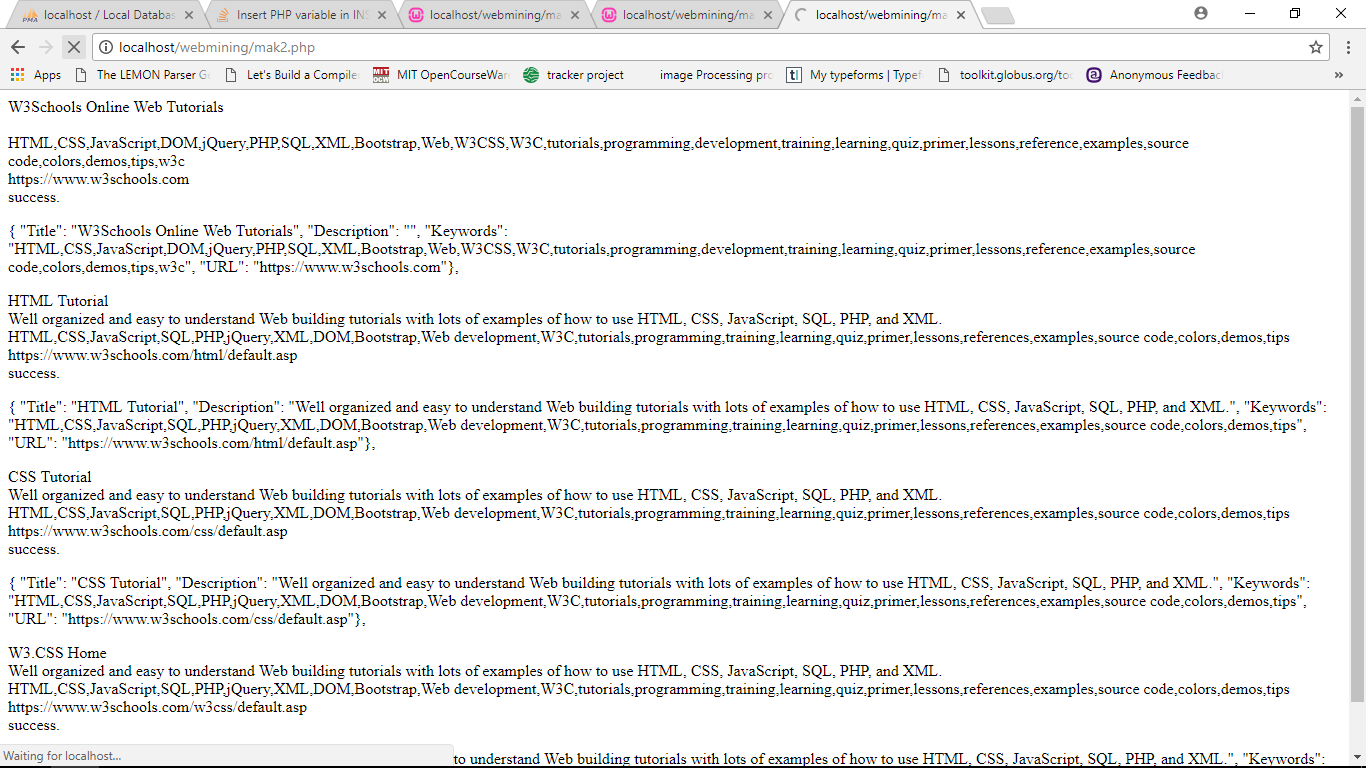
}

follow\_links($start);

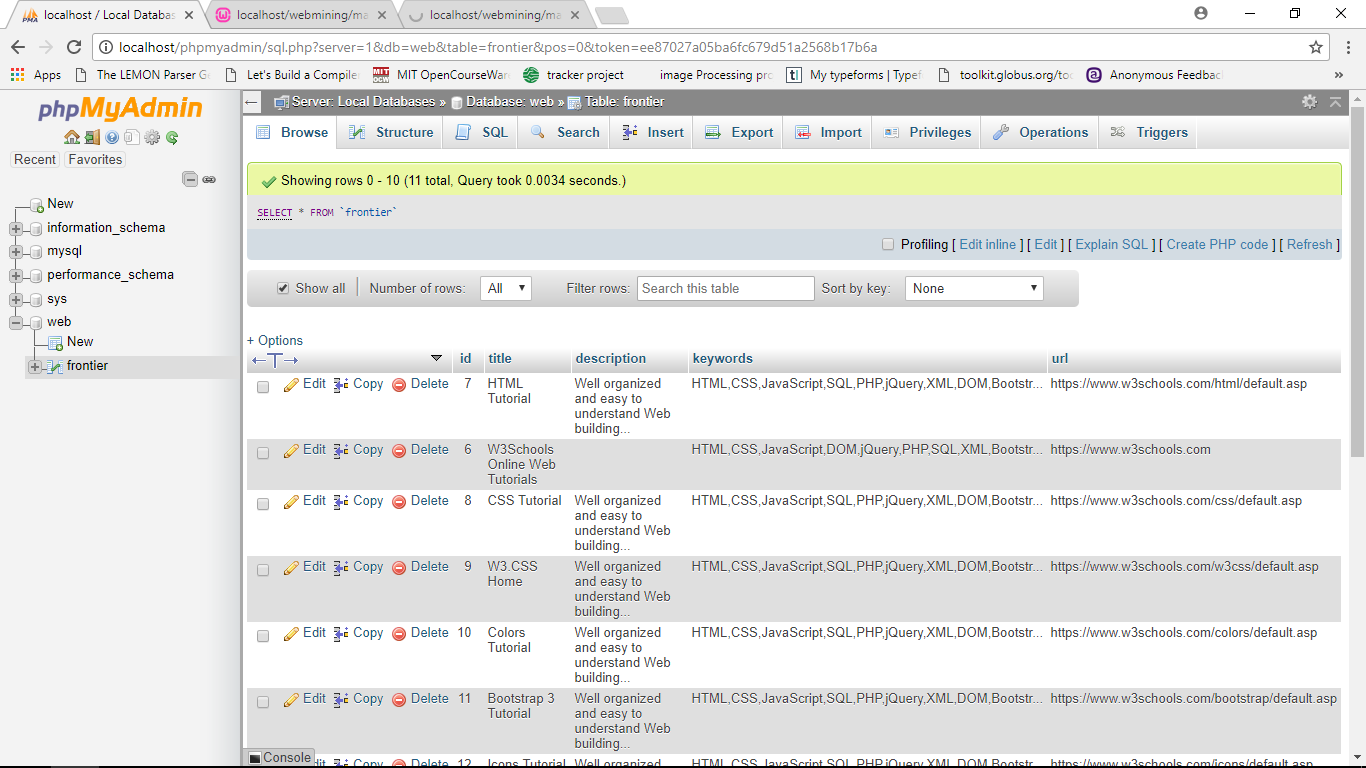
?>

The code for Tropical Crawler ends here. The databases displaying URL details can be seen in the output below, and the sets of crawled and not crawled pages can be seen below. .

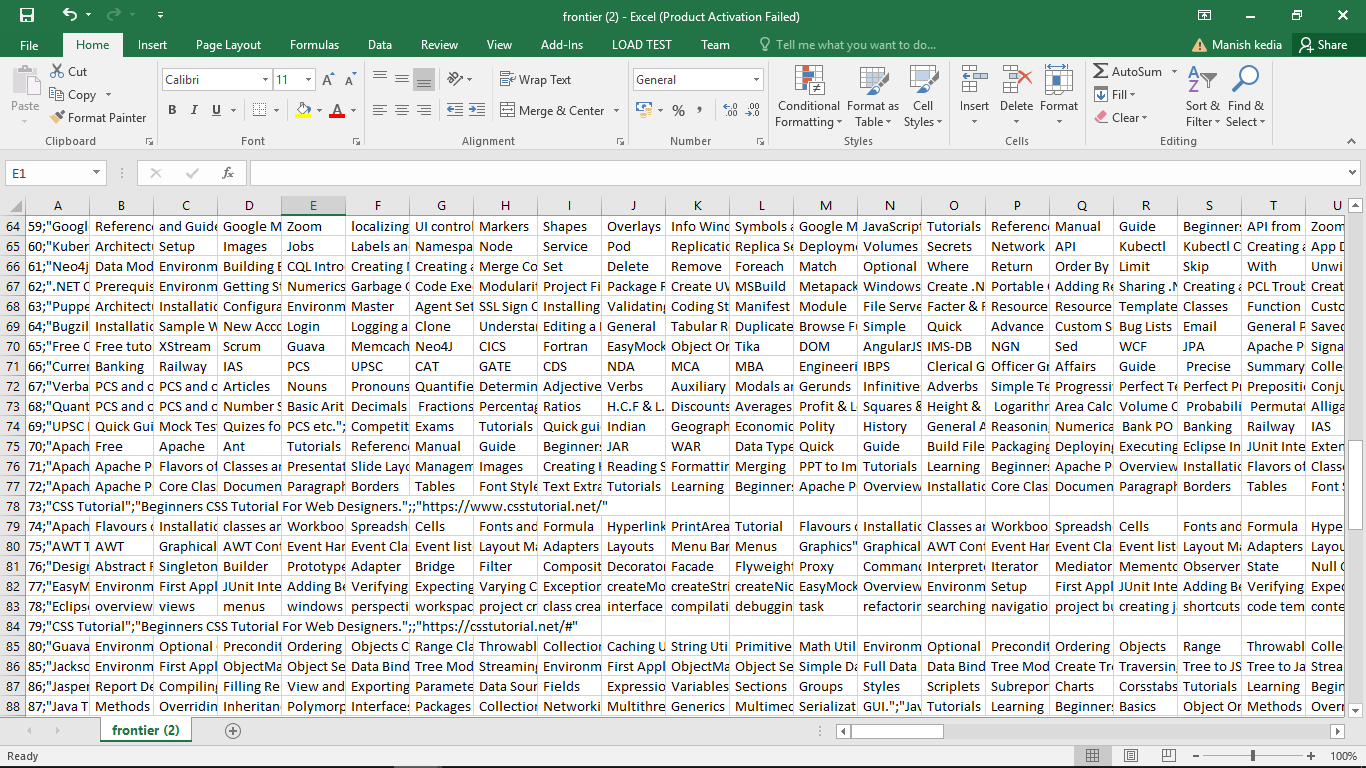
**OUTPUT OF PARSING:**

**

**FRONTIER TABLE:**

****

**EXTRACTED DATA FROM THE DATABASE IN AN EXCEL SHEET**

****

**FRONT END:**

This is the part that reads users query and process it to show output pages. The code is given below:

**Manish.html**

<!DOCTYPE html>

<!--

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

-->

<html>

<head>

<title>CHAOS</title>

</head>

<body style="margin-left: 25px;">

<h1>

<center>CHAOS SEARCH ENGINE</center>

</h1>

<form action="tables.php" method="get">

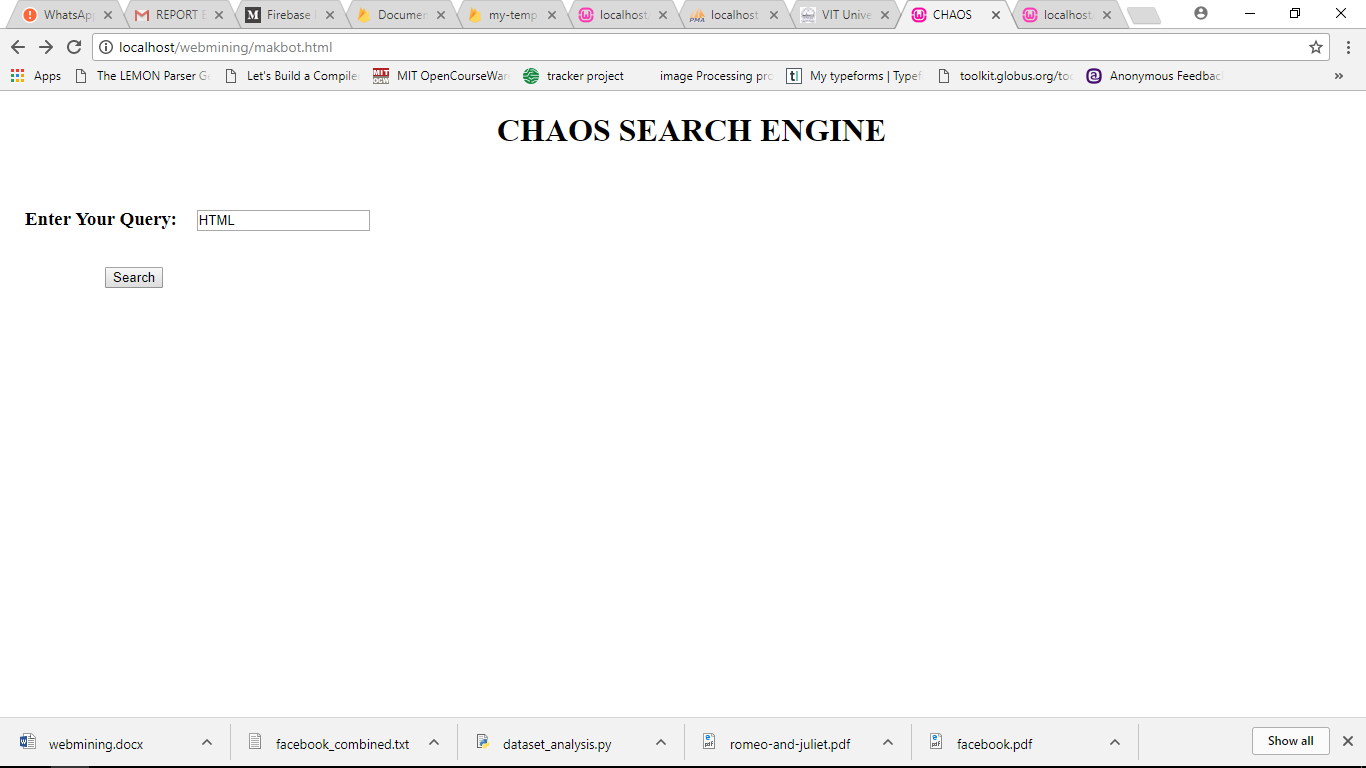
<br/><h3>Enter Your Query:<input type="text" name="query" id="query" style="margin-left: 20px;"></h3><br>

<input type="submit" value="Search" style="margin-left: 80px;">

</form>

</body>

</html>



**SEARCH RESULTS:**

The search results are shown on the next page. The code and the output for it is given below:

**Tables.php**

<?php

$conn = mysqli\_connect("localhost","root","","web");

if (!$conn)

{

die('Could not connect: ' . mysqli\_error());

echo "not connected";

}

$a = $\_GET['query'];

echo $a;

$query2 = "SELECT \* FROM frontier WHERE keywords LIKE '%$a%' and title like '%$a%';";

$run = mysqli\_query($conn,$query2);

if (!$run)

{

echo "This project is doomed";

}

$i=0;

$j=0;

while($result = mysqli\_fetch\_assoc($run))

{

$title[] = $result['title'];

$description[] = $result['description'];

$keywords[] = $result['keywords'];

$url[] = $result['url'];

$i++;

}

//for debugging and not for normal use

// print\_r($title);

// echo "\n";

// print\_r($description);

// echo "\n";

// print\_r($keywords);

// echo "\n";

// print\_r($i);

// print\_r($url);

// echo "\n";

mysqli\_close($conn);

?>

<html>

<head>

<title></title>

<script type="text/javascript">

</script>

<style type="text/css">

table, th, td {

border: 1px solid black;

}

th,td{

padding-left: 5px;

padding-right: 5px;

}

</style>

</head>

<body>

<h1>THE RESULTS OF YOUR QUERY IS GIVEN BELOW: </h1>

<table style="margin-top: 20px; margin-left: 20px; ">

<tr>

<th>Title</th>

<th>Description</th>

<th>Keywords</th>

<th>URL</th>

</tr>

<?php for($j=1; $j<=$i; $j++)

{;

?>

<tr>

<td><?php echo $title[$j-1]; ?></td>

<td><?php echo $description[$j-1]; ?></td>

<td><?php echo $keywords[$j-1]; ?></td>

<td><a href="<?php echo $url[$j-1]; ?>" target="\_blank"><?php echo $url[$j-1]; ?></a></td>

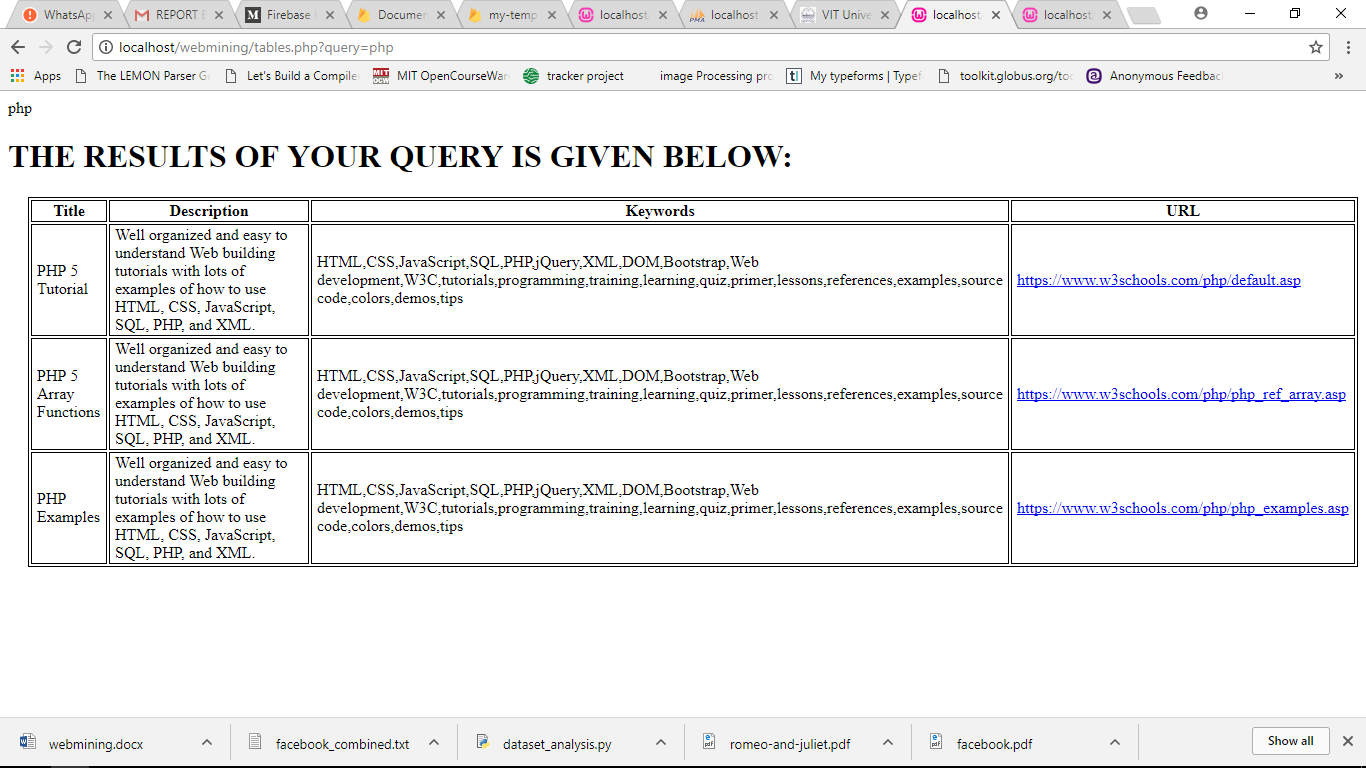
</tr>

<?php }; ?>

</table>

</body>

</html>

****

**GITHUB REPOSITORY LINK:**

[**https://github.com/hkedia321/topical-search-engine**](https://github.com/hkedia321/topical-search-engine)