# Project Report: Web Crawler and Search

## Crawler (crawler.php) Overview:

### Seed URL and Depth:

* The script starts with a seed URL ($url) and a specified depth ($depth) for crawling.

### Output Directory Cleaning:

* Clears the output directory ($output\_dir) before starting a new crawl. Existing files are deleted using unlink within a loop.

### Robots.txt Processing:

* Parses the seed URL to construct the robots.txt URL and fetches its content using cURL.
* Disallowed URLs are extracted from the robots.txt content.

### URL Queue Initialization:

* Initializes variables such as $orig\_url, $urls\_to\_scrap, and $completed\_urls for managing the crawling process.

### Crawling Loop:

* Initiates a loop that continues until either the specified depth is reached or there are no more URLs to crawl.
* Each iteration involves loading a page using cURL, handling errors, and checking for "PAGE NOT FOUND" messages.

### HTML Processing:

* Utilizes PHP's DOMDocument for HTML parsing.
* Saves the HTML content to a file in the output directory based on the current depth.

### URL Extraction and Filtering:

* Extracts URLs from anchor tags (<a>).
* Filters out URLs starting with "#" and adds the base URL to those starting with "/".
* Validates URLs and ensures they have not been crawled or disallowed.

### URL Queue Update:

* Updates the URL queue with newly extracted and filtered URLs.
* Decrements the depth and adds the current URL to the completed URLs list.

## Search (search.php) Overview:

### Query and Output Configuration:

* Specifies the search query ($query) and the number of results to display ($n).

### DOMObject Initialization:

* Loads HTML files from the output directory into DOMDocument objects.

### Data Extraction:

* Extracts textual content from various HTML elements, including title, headings, paragraphs, spans, anchors, list items, table cells, labels, and buttons.

### Vector Representation and Scoring:

* Converts the query and document data into vectors.
* Calculates the cosine similarity scores between the query and each document.

### Result Sorting and Output:

* Sorts the documents based on similarity scores and selects the top N results.
* Highlights the query in the output for better visibility.

## Bonus Features:

### Filtering (Crawler):

* Implements robots.txt compliance by checking and respecting disallowed paths during crawling.

### Error Handling (Crawler):

* Uses try-catch blocks for error handling during cURL requests.

### Search Result Highlighting (Search):

* Highlights the search query in the displayed results.

## Areas for Improvement:

1. **Logging and Reporting:**
   1. Integrate a logging mechanism to capture and report errors during crawling and searching.
2. **Concurrency:**
   1. Explore options for parallelizing or optimizing the crawling process for better performance.
3. **User Interface:**
   1. Consider developing a simple user interface for better interaction and visualization of search results.
4. **Advanced Search Features:**
   1. Enhance the search functionality to support more advanced features like case sensitivity, regular expressions, or searching within specific HTML elements.
5. **Persistent Storage:**
   1. While the crawler saves data to files, consider utilizing a more robust persistent storage solution, such as a database, for scalability.

This report provides an in-depth analysis of the provided web crawler and search scripts, highlighting their functionalities, bonus features, and suggesting areas for improvement and future development.