WebScour: A Distributed Web Crawler and Search Engine

1.Introducion To Project:-

WebScore is a distributed web crawler and search engine that collects web pages from the internet, indexes their content, and provides ranked search results. It uses multiple crawler nodes working in parallel to improve speed, scalability, and efficiency in retrieving relevant information from large-scale web data.

2.website ->Store Data->Uses for Search

1.Purpose-To collect, index, and search web pages efficiently.
2.WebCrawler-Automatically visits websites and downloads web pages.
3.Distributed System – Multiple crawler nodes work in parallel.
4.Speed -Parallel crawling increases data collection speed.
5.Scalability-Can handle large numbers of web pages.
6.Data Storage-Stores page content and metadata in a database.
7. Indexing - Organizes web data for fast searching.
8.SearchEngine- Processes user queries and retrieves results.
9.Outcome – Provides accurate and fast search results.

3.Diff Between Web Crawler and Web Scraper:-

Web Crawler = Finds pages
Web Scraper = Extracts data from pages

Web Crawler	Web Scraper
1.Automatically Browses websites by following Links.	Extracts specific data from web pages.
2. Used to discover and index pages.	Used to collect targeted information.

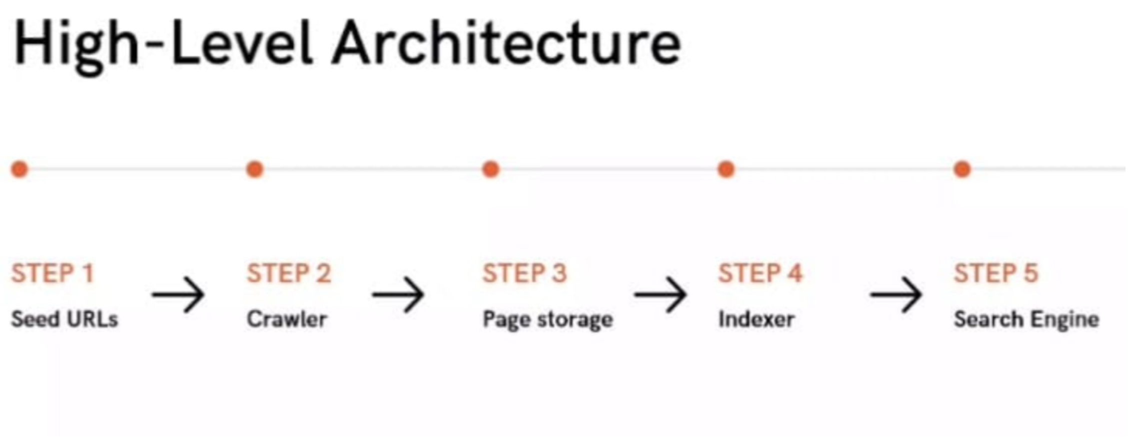
3.Covers Large number of Web Pages	Focuses on selected pages or data Files.
4.Works continuously and sytematically	Runs on demand for data extraction
5.Example:- Google bot	Example:- Product price scraper.

Web Crawler Example

1.Google Bot visits a website.
2.It follows links from one page to another.
3.It discovers and indexes thousands of web pages.

Web Scraper Example:

1.A price comparison tool visits an online shopping site.
2.It extracts product names, prices, and ratings.
3.It stores this data for analysis.



Step 1: Seed URLs

These are the starting web addresses.
Example:
https://example.com, https://wikipedia.org
The system begins crawling from these URLs.

Step 2: Crawler

The crawler (bot/spider) visits each seed URL.
Downloads web pages
Extracts links from pages
Follows new links automatically
Purpose: Collect web pages from the internet

Step 3: Page Storage

All downloaded pages are stored here.
Stores:
HTML content
Page metadata (URL, time, size)
Acts like a database of crawled pages

Step 4: Indexer

The indexer processes stored pages.
Extracts keywords
Removes stop words
Creates an index (word → page mapping)
Purpose: Make searching fast and efficient

Step 5: Search Engine

This is the user-facing part.
User enters a query (e.g., “web crawler”).
The search engine:
Looks into the index
Finds relevant pages
Ranks results

Simple Flow

Seed URLs → Crawler → Page Storage → Indexer → Search Engine

Real-World Example

Seed URL: google.com

Crawler: Googlebot

Page Storage: Google servers

Indexer: Google indexing system

Search Engine: Google Search

6.What Does Problem webscour Solve?

1. Problem it addresses
Huge volume of web data: The web has billions of pages. Finding relevant information manually is impossible.
Efficient search: Users need quick and accurate search results for their queries.
Organizing web content: Web pages are unstructured and scattered. There needs to be a system to categorize and rank them.

2. How WebScore solves it
1. Web crawling: Automatically collects web pages from different websites.
2. Indexing: Stores web pages in a structured format to allow fast search.
3. Ranking (Scoring): Assigns scores to pages based on relevance, popularity, or other criteria (like Google PageRank). This helps show the most useful results first.
4. Distributed search: Can handle large-scale data by using multiple machines to crawl, store, and rank pages efficiently.

7.What Does "Distributed"Mean Here ?

More data = Morecrawling works
one crawler running on one machine= slow
Distributed = work is shared or can be shared:
    Multiple crawler workers
    Possibly separate components:- crawler,indexer,search

8.Quick Python Basics Recap

Variable & Data Type

A variable is a name used to store data that can change during program execution.

```
x = 10
name = "CTF"
```

A data type specifies the kind of data a variable can store.

```
age = 20         # Integer
price = 99.99    # Float
name = "Alex"    # String
is_active = True  # Boolean
```

Functions(def , Parameters, Return)

A function is a block of reusable code that performs a specific task

```
def function_name():
 code
```

Parameters

Parameters are variables listed inside the parentheses.
They receive values when the function is called.

Return

The return statement sends a value back to the caller.
It ends the function execution.

Example:
```
def add(a, b):
   return a + b
```

Key Points

def → defines the function
Parameters → inputs to the function
return → outputs from the function

List & Dictionaries(With small example)
List → Ordered collection (uses index)
Dictionary → Uses keys instead of index

9. Functional Requirements

These define what the system should do:

1. Web Crawling:Automatically browse and collect web pages from multiple websites.

2. Data Storage/Indexing:

Store collected pages in a structured format for quick retrieval.

3. Search Functionality:

Allow users to enter queries and retrieve relevant pages.

4. Ranking/Scoring:

Score web pages based on relevance, keywords, or popularity.

10 Non-Functional Requirements

These define how well the system should perform:

1. Performance:

Fast crawling and search response time, even for large datasets.

2. Scalability:

Able to handle growing amounts of web data efficiently.

3. Reliability:

Should work continuously without crashing or losing data.

4. Usability:

Easy-to-use interface for users to search and view results.

11.Technology Stack

1. Frontend (User Interface)

HTML/CSS/JavaScript: Basic interface for search input and displaying results.

React.js or Angular (optional): For a more interactive frontend.

2. Backend (Server & Logic)

Node.js / Python / Java: Handles requests, runs search queries, and communicates with the database.

Flask / Django (if Python used): For building web APIs.

3. Web Crawling

Python libraries: Scrapy, BeautifulSoup, Requests

Java libraries: Jsoup, Apache Nutch

4. Database

Relational Database: MySQL, PostgreSQL (for structured storage)

NoSQL Database: MongoDB, Elasticsearch (for fast search and indexing)

Conclusion:-

The WebScore project successfully addresses the challenge of efficiently finding and ranking relevant web information from the vast amount of online data. By integrating web crawling, indexing, and search ranking, it enables users to quickly access accurate and relevant results. The system demonstrates scalability, reliability, and improved user experience, making it a practical solution for managing large-scale web data.