THE INSTITUTE OF FINANCE MANAGEMENT

FACULTY OF COMPUTING AND MATHEMATICS

DEPARTMENT OF COMPUTER SCIENCE

CSU/ITU07315: Web Technologies and Internet Programming

Group Assignment

Overview

Google is a leading global search engine platform, handling over 8.5 billion search requests per day, approximately 99,000 searches per second, and more than 5 trillion searches per year. The search process is done using a well-structured and fine-tuned searching algorithm. Search results are ranked using hundreds of factors, including keyword relevance, links, location, and freshness. The page that presents search results is generated in a fraction of a second, with the most helpful information at the top.

Similar in spirit to the Google search engine, you are supposed to build a minimalistic search engine using HTML/CSS, PHP, and MySQL/PostgreSQL. The application will consist of two main pages:

- A clean home page with a centered search box
- A search results page that displays matching records and also contains a search box at the top (so users can search again without going back)

You will be provided the required two pages (static pages) and a dataset of ~10,000 records stored in an associative array format. You are supposed to write an insertion script to populate the database and implement basic but functional search logic.

Objectives

By the end of this assignment, each group should be able to:

- Design and implement a simple, working two-page search application, search and ranking algorithm
- Populate a database with a large sample dataset using pure PHP
- Implement a usable full-text-like search using LIKE (or ILIKE in PostgreSQL) with support for multiple keywords

- Write clean, readable, and reasonably secure PHP code (basic input sanitization, prepared statements encouraged)
- Work collaboratively in a team of 10 students and divide responsibilities effectively
- Demonstrate incremental progress during tutorial sessions

The goal is **not** to build Google, but to understand the core mechanics of accepting a query, searching a database, and presenting ranked results in a user-friendly way.

Technical Requirements

i. Database Structure

Create one main table named **search_items** with at least the following columns:

SQL

```sql
 CREATE TABLE search_items (
id                BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
title             VARCHAR(255) NOT NULL,
description       TEXT,
page_name         VARCHAR(255) NOT NULL,
page_fav_icon_path  VARCHAR(255) NOT NULL,
page_url          VARCHAR(500) DEFAULT NULL,
created_at        DATETIME DEFAULT CURRENT_TIMESTAMP,
FULLTEXT INDEX ft_search (title, description)
);
```

- PostgreSQL users: use SERIAL instead of AUTO_INCREMENT and ILIKE instead of case-sensitive LIKE

ii. Sample Data

You will be provided with a PHP array containing ≈10,000 records

Example structure of each record (associative array):

PHP

```php
$sampleData = [
  [
'title'        => 'PHP and MySQL Web Development',
'description' => 'A comprehensive guide to building dynamic websites...',
'page_name'    => 'PHP and MySQL Web Development',
'page_fav_icon_path' => '/images/php-mysql-favicon.ico',
'page_url'     => 'https://example.com/books/php-mysql',
'created_at'   => '2024-01-01 12:00:00'
],
// ... 9,999 more records ...
 ];
```

Write a separate PHP script (insert_data.php) that:

- Connects to your database
- Loops through the array
- Inserts all records using prepared statements (strongly recommended)
- Can be safely re-run (e.g., truncate table first or use INSERT IGNORE)

iii. Application Pages & Features

A. Home page (index.html) - Provided

- Beautiful, centered search box (large input + submit button)
- Minimal design — focus on the search bar (use CSS to center vertically & horizontally)
- Optional: logo

B. Search results page (results.html) - Provided

- Search box at the top (pre-filled with the previous query)
- Results displayed below in a clean list or cards
- Each result should show: page name, fav icon, title**,** description snippet**,** link(fake URL as a clickable link)

- Shows the total number of results found
- Time taken to search and produce results (in seconds)
- Basic pagination (e.g., 20 results per page)
- Highlight matching keywords in title/description

C. Basic Search Logic

- Accept GET or POST method for the search query
- Split query into words (explode on space)
- Search both the title and description columns
- Use LIKE '%word%' for each word (AND logic — record must match all words)
- Order results by relevance (e.g., higher popularity first, or number of matches)
- Sanitize/escape input to prevent SQL injection
- Show "No results found" message when appropriate

D. Other Internal Working Features (at a minimum include)

- Database connection in a separate file (config.php / db.php)
- Basic error handling (connection errors, empty query, etc.)
- Basic input validation/trimming
- Clean URLs if using GET (no ugly long query strings if possible)

Rules

- Groups of exactly 10 students (no exceptions without tutor approval)
- No usage of AI code generators (ChatGPT, Copilot, Claude, Gemini, etc.) for writing PHP/SQL/HTML/CSS code — violation = 0 for the group
- You may use AI only to explain concepts or debug small errors — not to generate code
- All group members must understand and be able to explain every file/line of the code
- Code must be hand-written or written collaboratively during meetings
- You are supposed to write vanilla HTML/CSS and PHP. No Framework
- No submission deadline — work is assessed progressively during tutorial sessions
- Final presentation/demo + code review will happen in the last 1–2 tutorial sessions