PERSONALISATION AND PRIVACY IN PROFILE BASED WEB SEARCH

A MINI PROJECT REPORT

Submitted by

SHAHAMA SHAHABUDEEN [REGISTER NO: 211420205138]

SOUNDARYA.S [REGISTER NO: 211420205149]

TEJASHWINI.V [REGISTER NO: 211420205166]

SUMITHA.S [REGISTER NO: 211420205155]

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, CHENNAI 600 123

(An Autonomous Institution)

ANNAUNIVERSITY: CHENNAI 600 025

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"PERSONALIZATION AND PRIVACY IN PROFILE BASED WEB SEARCH"** is the bonafide work of **"SHAHAMA SHAHABUDEEN (211420205138), TEJASHWINI.V (211420205166), SOUNDARYA.S (211420205149), SUMITHA.S (211420205155)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. HELDA MERCY M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

SIGNATURE

Mrs. MUTHULAKSHMI M.Tech.,(Ph.D)
ASSOCIATE PROFESSOR (SUPERVISOR)
Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

DECLARATION

I hereby declare that the project report entitled "**PERSONALISATION AND PRIVACY IN PROFILE BASED WEB SEARCH ”**which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor of Technology in Information Technology' in **Panimalar Engineering College, An Autonomous institution Affiliated to Anna University- Chennai** is the result of the project carried out by me under the guidance and supervision of **Mrs. K.MUTHULAKSHMI, M.TECH.,(Ph.D) Associate Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

(Shahama Shahabudeen)

(Soundarya.S)

Date: **(Tejashwini.V)**
Place**:** Chennai

(Sumitha.S)

It is certified that this project has been prepared and submitted under my guidance.

Date: (**Mrs. K.MUTHULAKSHMI M.Tech.,(Ph.D)**

Place: Chennai (Associate Professor/ IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co- operation and support from many, for successful completion. We wish to express our sincere thanks to all thosewho were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYA SREE SAKTHIKUMAR., B.E., M.B.A.,Ph.D** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our staff in charge, **Mrs.K. MUTHULAKSHMI M.Tech.,(Ph.D)** Associate Professor, Department of Information Technologyfor her guidance throughout the course of our project. Last, we thank our parents and friends forproviding their extensive moral support and encouragement during the course of the project.

ABSTRACT

A search technique to enhance the efficacy and quality of web searches is personalized search results. lately it has received a lot of interest. Personalizing results for searches to make them more appropriate and suited to the client's preferences is the fundamental objective of personalized web search. But for personalization to be beneficial, user data—whether private or public—must be gathered and aggregated. The effectiveness of the customized online search system is impacted by users' reluctance to divulge certain confidential data when using search engines. This includes a poll on user choices in personalized web searches that are modelled as individual profiles. This study also discusses a personalized web search system that flexibly generalizes individuals while taking into account viewer-specified confidentiality constraints. For a given query, a personalized Web search can provide different search results for different users or organize search results differently for each user, based upon their interests, preferences, and information needs. Personalized web search differs from generic web search, which returns identical research results to all users for identical queries, regardless of varied user interests and information needs. Different users have different backgrounds and interests. They may have completely different information needs and goals when providing exactly the same query., since it can provide different search results based upon the preferences and information needs of users. It exploits user information and search context in learning to which sense a query refers. To provide personalized search results to users, personalized web search maintains a user profile for each individual. A user profile stores approximations of user tastes, interests and preferences.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT:

Several concerns have been brought up regarding personalized search. It decreases the likelihood of finding new information by biasing search results towards what the user has already found. It introduces potential privacy problems in which a user may not be aware that their search results are personalized for them, and wonder why the things that they are interested in have become so relevant. The methods of personalization, and how useful it is to "promote" certain results which have been showing up regularly in searches by like-minded individuals in the same community Personalized search results are a search technique to improve the effectiveness and quality of web searches. It has attracted a lot of interest recently. The main goal of personalized online search is to tailor search results so that they are more relevant and tailored to the client's tastes. However, user data, whether private or public, must be acquired and aggregated for personalization to be useful. Users' unwillingness to provide some sensitive information when using search engines has an impact on the effectiveness of the tailored online search system. This essay comprises a survey on user preferences for customized web searches that are created using personal data. This study also presents a customized web search system that adaptably generalizes people while taking into consideration confidentiality restrictions provided by viewers.

1.2 NEED FOR THE PROJECT

- **To maintain A private profile**
- **To Provide privacy in web search**
- **Maintaining Security in Search Engine**

Personalization in and of itself is not a goal, in fact, it’s a tool to reach the goal of relevancy, accuracy, and speed. Personalization means looking at user-specific signals to make your product or content more real When it comes to your internal site search, personalized search results are not just based on traditional ranking factors (relevancy, boosting of content). They also take the information the search engine has about the user into account – including their location, language, search history, or device. Personalized search results are more accurate and relevant to the particular user, allowing them to find the information they need immediately. Overall, the experience for the user is better. That can lead to higher conversion rates, longer visit durations, lower bounce rates, and improved brand reputation. For example, streaming services such as Spotify promotes music related to the one you listened to in

the past, while demoting others that you are unlikely to listen to. When it comes to your internal site search, personalized search results are not just based on traditional ranking factors (relevancy, boosting of content). They also take the information the search engine has about the user into account – including their location, language, search history, or device. Personalized search results are more accurate and relevant to the particular user, allowing them to find the information they need immediately.

2.3 OBJECTIVE OF THE PROJECT

The quality and efficacy of web searches can be increased by using personalized search results. Recently, there has been a lot of interest in it. Personalizing search results online is mostly done to make them more relevant and suited to the client's preferences. Personalization, whether it be public or private, needs to be collected and aggregated in order to be effective. The efficacy of the customized online search system is impacted by users' refusal to reveal some personal information when using search engines. This essay includes a survey on users' preferences for personalized web searches made with their personal information. This study also offers a personalized web search engine that generalizes persons well while taking confidentiality limits into account.

2.4 SCOPE OF THE PROJECT

The algorithms and the data sets adopted are intended to be popular and easily accessible for anyone interested in this research area. However, it would be of greater value evaluating the performance of the measures on larger test-beds. It would be interesting to investigate the effectiveness and to try resist adversaries with border background knowledge such as richer relationship among topics such as exclusiveness, sequentially and so on or capability to capture series of queries from victim. It will be more interesting to seek more sophisticated method to build the user profile and better metrics to predict the performance of UPS.

ADVANTAGES

- Privacy Management
- Safe And Secure
- Predictive analysis
- Improved safety

CHAPTER 2

2. LITERATURE SURVEY

Chen et al. Personalized learning has been a topic of research for a long time. However, around 2008, personalized learning started to draw more attention and take on a transformed meaning as seen. However, we believe the variety of terms that have been used for personalized learning seems to be an obstacle to the progress of personalized learning theories and research. Although there exists an abundance of the resources/studies on personalized learning, not having a readily agreed-upon term of personalized learning might be the obstacle in research progress on personalized learning. In response to this need, this paper is focused on analyzing the terms that have been used for personalized learning. A distinctly personalized learning approach can help the educational researchers to build up research on previous data, instead of trying to start new research from scratch each time. This paper will present a research-based framework for personalized learning and discuss future research directions, issues, and challenges through an in-depth analysis of the definitions and terms used for personalized learning. Personalized learning has existed for hundreds of years in the form of apprenticeship and mentoring. As educational technologies began to mature in the last half of the previous century, personalized learning took the form of intelligent tutoring systems. In this century, big data and learning analytics are poised to transform personalized learning once again. Learning has been characterized as a stable and persistent change in what a person knows and can do. Personalized learning is a complex activity approach that is the product of self-organization or learning and customized instruction that considers individual needs and goals. Personalized learning can be an efficient approach that can increase motivation, engagement and understanding, maximizing learner satisfaction, learning efficiency, and learning effectiveness. However, while such personalized learning is now possible, it remains as one of the biggest challenges in modern educational systems. In this paper a review of progress in personalized learning using current technologies is provided. The emphasis is on the characteristics of personalized learning that need to be taken into consideration to have a well-developed concept of personalized learning. We started with the definition of personalized learning suggested by Spector (2014, 2018) and others that are discussed below, which requires a digital learning environment to be classified as a personalized learning environment to be adaptive to individual knowledge, experience and interests and to be effective and efficient in supporting and promoting desired learning outcomes. These characteristics are those which are typically discussed in the research community although we found it challenging to find a sufficient number of published cases that reported effect sizes and details of the sample in order to conduct a formal meta-analysis. Lacking those cases suggests that personalized learning in the digital era is still in its infancy. As a result, we conducted a more informal albeit systematic review of published research on personalized learning.

Furthermore, we, along with many educational technologists, believe an efficient personalized learning approach can increase learners' motivation and engagement in learning activities so that improved learning results. While that outcome now seems achievable, it remains a largely unrealized

opportunity according to this research review. Truong stated that providing the same content to students with different qualifications and personal traits and having different interests and needs is not considered adequate anymore when learning can now be personalized. Miliban promoted personalized learning to be the solution to tailoring the learning according to individuals' needs and prior experience so as to allow everyone to reach their maximum potential through customized instruction. The customized instruction that includes what is taught, how it is taught, and the pace at which it is taught. This allows learning to meet individual needs, interests and circumstances which can be quite diverse. Furthermore, FitzGerald et al. (2018) pointed out the personalization of learning is now a recurring trend across government agencies, popular media, conferences, research papers, and technological innovations.

Personalized learning is in demand (Chen, 2012) due to new technologies involving big data and learning analytics. It should be tailored to and continuously modified to an individual learner's conditions, abilities, preferences, background knowledge, interests, and goals and adaptable to the learner's evolving skills and knowledge. Today's personalized learning theories are inspired by educational philosophy from the progressive era in the previous century, especially John Dewe emphasis on experiential, learner-centered learning, social learning, extension of the curriculum, and fitting for a changing world. McCombs claimed that a learner-centered environment develops as it considers learners' unique characteristics using the best knowledge of teaching and learning which are available. Furthermore, claimed that individualized learning is a tool to facilitate learner-centered education. FitzGerald et al. (2018) pointed out that personalization is a crucial topic of current interest in technology-oriented learning design and discussion for government policymakers, but less so in educational research. This might be a good explanation of disunity of personalized learning approaches. Individualized instruction is one of the terms that are often used to talk about the specific needs and goals of individuals to be addressed during instruction. This notion has evolved from one-on-one human tutoring. It is not agreed upon whether individualization is a component of personalized learning or another term that can be used in place of personalized learning.

CHAPTER 3

SYSTEM DESIGN

3.1. REFERENCE ARCHITECTURE OF PROPOSED SYSTEM DESIGN

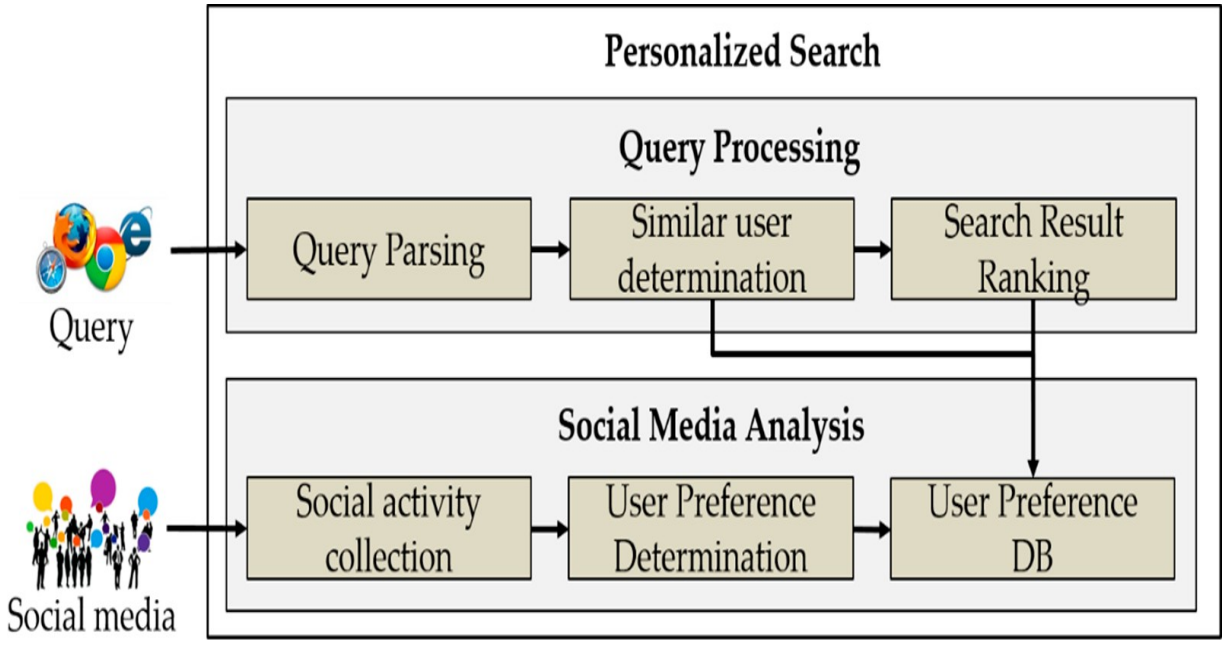


Figure 3.1. Reference architecture of the proposed search method

Web search engines provide users with a huge number of results for a submitted query based on the User Preference Determination. However, not all returned results are relevant to the user's needs. Personalized search aims at solving this problem by modeling search interests of the user in a profile and exploiting it to improve the search process. One of the challenges in search personalization. It is processed by analyzing the users interest based on the users search in the search engine. It contains two indexes in which the My index states the most related content the preferred search and the Synched Index provides the analysis of the previous queries and provides the preferred or the required results based on the users' search.

3.2. SYSTEM ARCHITECTURE:

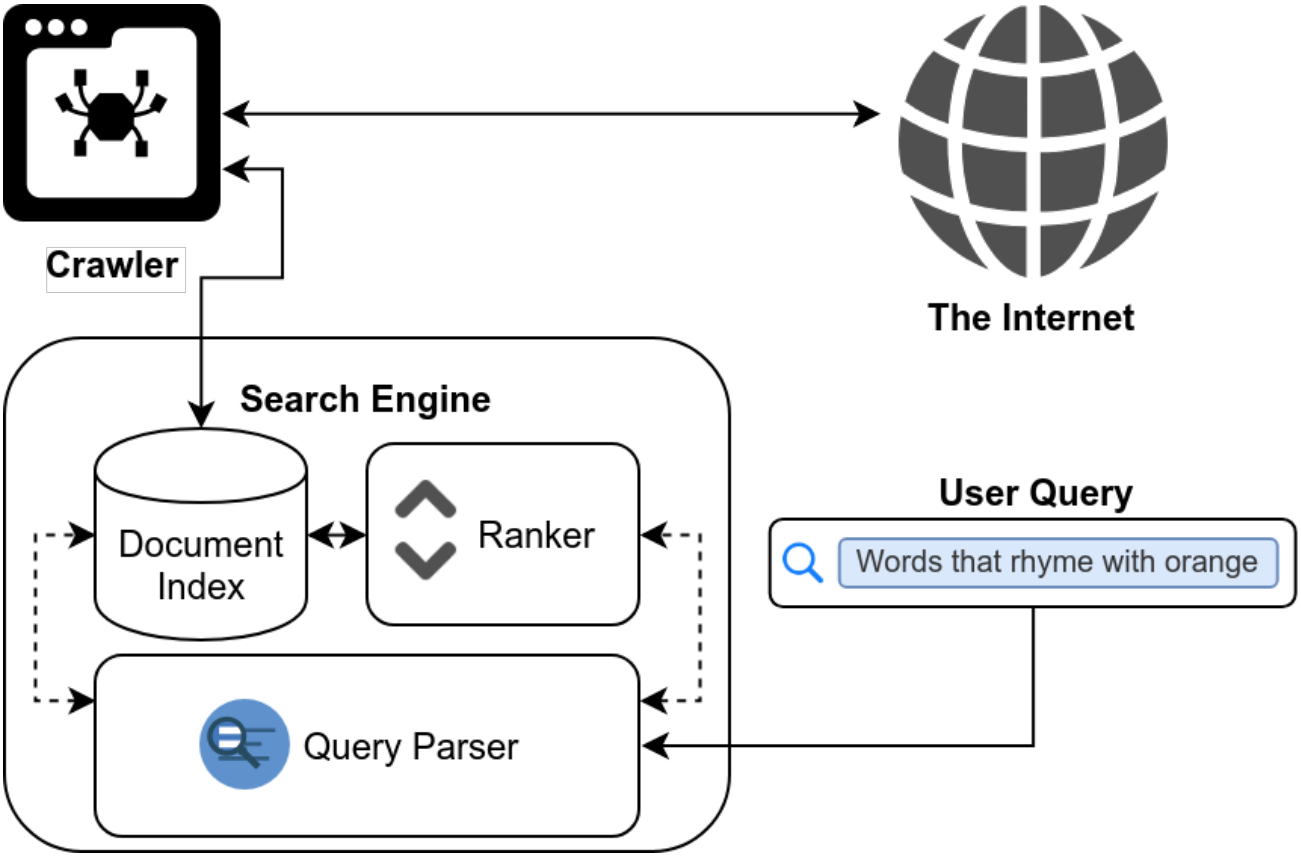


Fig 3.2 System Architecture

The user provides such a Query based on the users interest It contains the search Engine that provides the recommendations based on the Document indexes. It ranks the Queries based on the Ranker . The Query Parser Parses all the Query given by the User and Processes through the Internet. A crawler mostly does what its name suggests. It visits pages, consumes their resources, proceeds to visit all the websites that they link to, and then repeats the cycle until a specified crawl depth is reached. The first column indicates the similarity/relevance score the search engine assigned to the document, which you can modify to your pleasing. You'll also notice that various comments (text following #s) are used to indicate each query and the URLs corresponding to each row.

3.3 DATA FLOW DIAGRAMS:

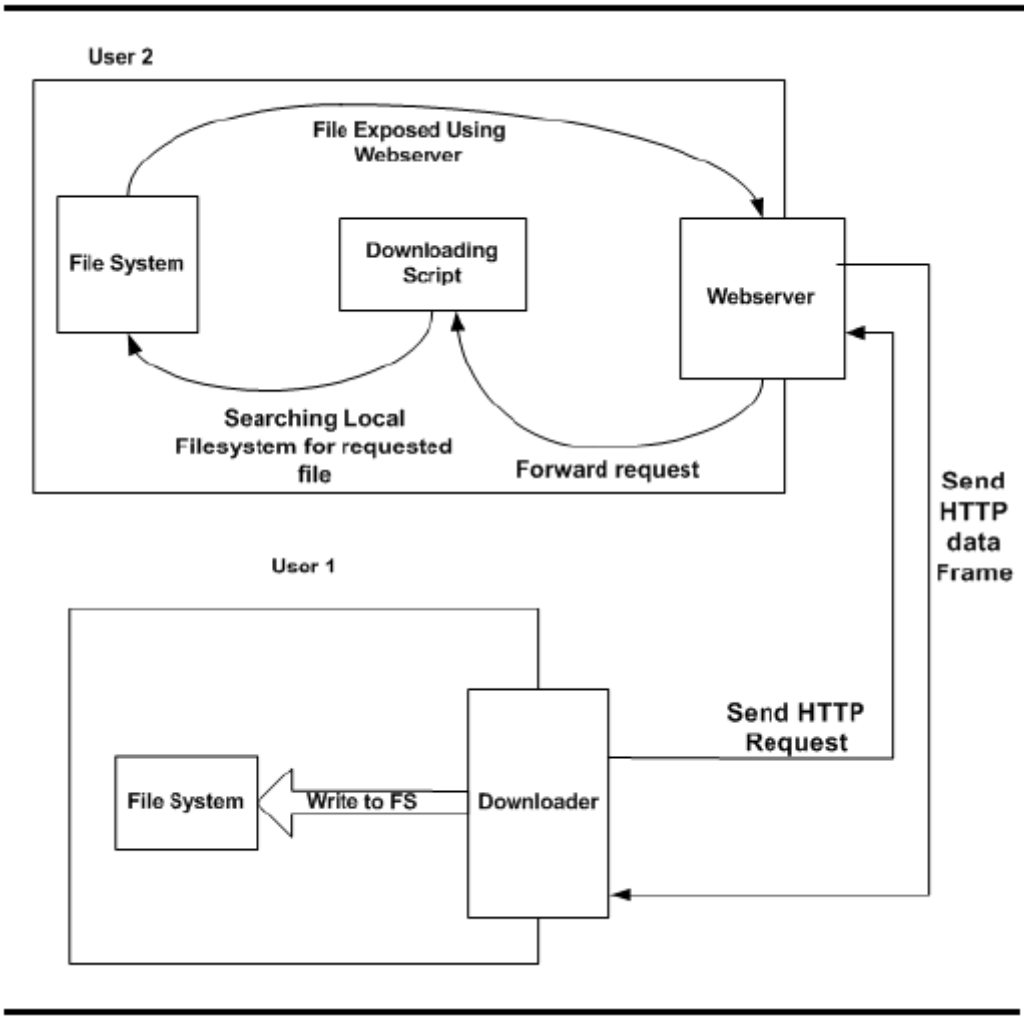


Fig 3.3 DFD diagram Level 1

The diagram above shows the basic interconnection between users and the main server. This diagram shows a breakdown of all the components that make up the overall system. All computers connected to the system are OAI compliant repositories which can be harvested by the harvester that is on the central server. These are the components that constitute the client system.

- **Local Search Engine**
- **Document Organizer**

- **Local Metadata Repository**
- **File Transfer Tool**
- **Citation Generation**

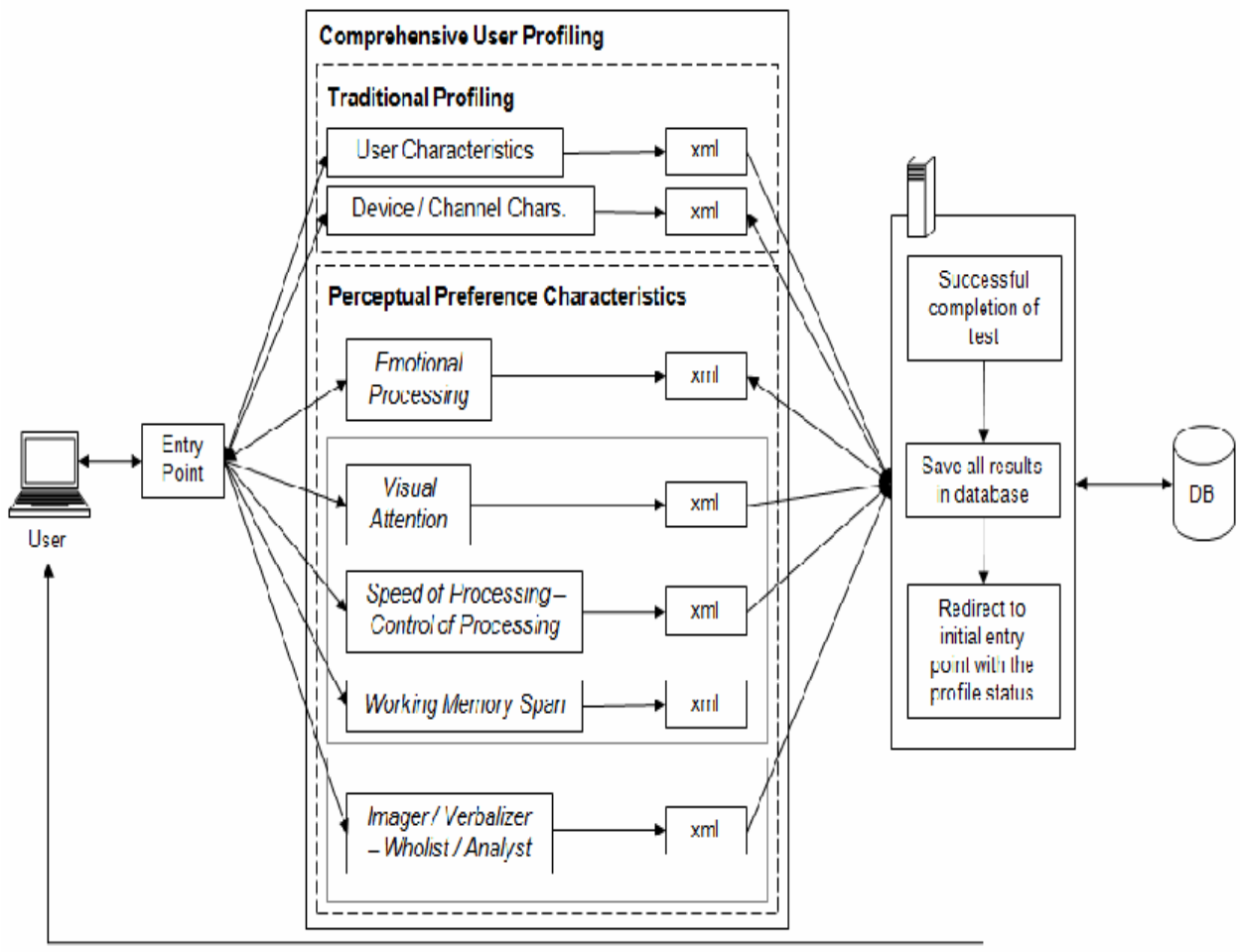


Fig 3.4 DFD Diagram Level 2

The downloading user sends an HTTP request to the host computer requesting a file. The host computer has a script that locates the file and sends it back using HTTP. HTTP status codes are used to communicate in case errors occur.

3.4 MODULE DESCRIPTION:

- **DataSet Preprocessing**
- **Query Submission and Query Retrieval**
- **Estimate Relevant Results**
- **Retrieve User Profile in Privacy Manner**

3.4.1 Dataset Preprocessing:

In this module, choose input dataset, The datas are loaded from the open-source Lucene browser which contains the necessary datasets in it. After loading the dataset into the database, we can view the dataset. By using the string-matching algorithm, we filter out unwanted values in the dataset and it has been preprocessed and store into the database.

3.4.1 Query Submission and Query Retrieval

Based on User In this module, user submits query. Based on the query, some relevant results have been displayed and also based on the submitted query some history results are displayed. Based on the query and already posted queries, we can calculate the similarity values between them. Based on the similarity values, we can estimate some results.

3.4.2 Estimate Relevant Results

In this module, user posts sub query also. Based on the query and sub query, we can estimate some results. Based on the relevant results and total amount of data in the dataset, we estimate the support values. In this, some history results are also found.

3.4.3 Retrieve User Profile in Privacy Manner

In this module, some adversaries to mine the history results means, only query time has been displayed. In this, other informations such as query, query results, username are not displayed by using the background knowledge. It helps in maintaining the privacy and security of the user in all means.

CHAPTER 4

4. REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS:

- Computer with ARM64 or x64 processor
- Processor: Minimum 1 GHz
- Memory (RAM): 4 GB
- Internet Connection

4.2 Software Requirements:

- **Front end** : HTML
- **Web application** : Visual Studio, Lucene open source
- **Back end** : C#, DotNET

4.2.1. FEATURES OF C#

About C#:

C# is pronounced "C-Sharp".

- It is an object-oriented programming language created by Microsoft that runs on the .NET Framework.
- C# has roots from the C family, and the language is close to other popular languages like C++ and Java.
- The first version was released in year 2002. The latest version, C# 11, was released in November 2022.

C# is used for:

- Mobile applications
- Desktop applications
- Web applications
- Web services
- Web sites

- Games
- VR
- Database applications
- And much, much more!

Methods and functions:

- Like C++, and unlike Java, C# programmers must use the scope modifier keyword virtual to allow methods to be by subclasses.
- Extension methods in C# allow programmers to use static methods as if they were methods from a class's method table, allowing programmers to add methods to an object that they feel should exist on that object and its derivatives.

Memory access:

- Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem by freeing the programmer of responsibility for releasing memory that is no longer needed in most cases. Code that retains references to objects longer than is required can still experience higher memory usage than necessary, however once the final reference to an object is released the memory is available for garbage collection.

Exception:

- **A** range of standard exceptions are available to programmers. Methods in standard libraries regularly throw system exceptions in some circumstances and the range of exceptions thrown is normally documented. Custom exception classes can be defined for classes allowing handling to be put in place for particular circumstances as needed. No implicit conversions occur between Booleans and integers, nor between enumeration members and integers.

4.2.2 Features of Dot NET

Introduction to Dot NET:

DotNET is an open-source platform for building desktop, web, and mobile applications that can run natively on any operating system. The .NET system includes tools, libraries, and languages that support modern, scalable, and high-performance software development. An active developer community maintains and supports the .NET platform.

In simple terms, the .NET platform is a software that can do these tasks:

- Translate .NET programming language code into instructions that a computing device can process.
- Provide utilities for efficient software development. For example, it can find the current time or print text on the screen.
- Define a set of data types to store information like text, numbers, and dates on the computer.

Implementation of Dot NET:

Various implementations of .NET allow .NET code to execute on different operating systems like Linux, macOS, Windows, iOS, Android, and many others. Dot NET Framework .NET has a modular and optimized architecture. Users can choose different components to meet their software development requirements.
These are the three main .NET components:

- .NET languages
- Application model frameworks
- .NET runtime

C#

C# is a simple, modern, and object-oriented programming language. With syntax similar to the C family of languages, C# is familiar to C, C++, Java, and JavaScript programmers

Visual Basic

Visual Basic is an object-oriented programming language developed by Microsoft. Using Visual Basic makes it fast and easy to create type-safe .NET apps. Type safety is the extent to which a programming language discourages or prevents logical coding errors.

.NET Runtime:

The .NET runtime, also called Common Language Runtime (CLR), compiles and executes .NET programs on different operating systems.

Just-in-time compilation:

The CLR compiles code as the developer writes it. During compilation, CLR translates the code into Common Intermediate Language (CIL). For example, code written in C# has English-like syntax and words. .NET compiles or translates this code into CIL. CIL code looks different because it is a lower-level machine code language.

Execution:

.NET runtime manages the execution of CIL code. CIL is cross-platform compatible, and any operating system can process it. Cross-platform compatibility refers to an application's ability to run on multiple different operating systems with minimal modifications. For example, an application in C# can run on Windows, Linux, or macOS without any code modifications. Such an application is called a cross-platform application.

.NET APPLICATION MODEL FRAMEWORKS:

The application model frameworks are a collection of developer tools and libraries that support fast and efficient .NET project development. Different frameworks exist for different types of applications.

Features of Lucene Open-source Database:

Apache Lucene is a free and open-source search engine software library, originally written in Java by Doug Cutting. It is supported by the Apache Software Foundation and is released under the Apache Software License**.** Lucene has also been used to implement recommendation systems. Lucene has been ported to other programming languages including Object Pascal, Perl, C#, C++, Python, Ruby and PHP.

Lucene: A full-text search library with core indexing and search services Competitive in engine performance, relevancy, and code maintenance

Scalable, High-Performance Indexing

- over 800GB/hour on modern hardware
- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed

Powerful, Accurate and Efficient Search Algorithms

- ranked searching -- best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- fielded searching (e.g. title, author, contents)
- nearest-neighbor search for high-dimensionality vectors
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching

Cross-Platform Solution

- Available as Open-Source software under the Apache License which lets you use Lucene in both commercial and Open Source programs
- 100%-pure Java
- Implementations in other programming languages available that are index-compatible

Searching and Indexing

Lucene is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead. This would be the equivalent of retrieving pages in a book related to a keyword by searching the index at the back of a book, as opposed to searching the words in each page of the book.

This type of index is called an inverted index, because it inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages).

Documents

- In Lucene, a Document is the unit of search and index.
- An index consists of one or more Documents.

Indexing involves adding Documents to an Index Writer, and searching involves retrieving Documents from an index via an Index Searcher. A Lucene Document doesn't necessarily have to be a document in the common English usage of the word. For example, if you're creating a Lucene index of a database table of users, then each user would be represented in the index as a Lucene Document.

Searching:

Searching requires an index to have already been built. It involves creating a Query (usually via a QueryParser) and handing this Query to an Index Searcher, which returns a list of Hits.

Queries:

Lucene has its own mini-language for performing searches. Read more about the Lucene Query Syntax.

The Lucene query language allows the user to specify which field(s) to search on, which fields to give more weight to (boosting), the ability to perform Boolean queries (AND, OR, NOT) and other functionality.

Indexing:

Simply put, Lucene uses an "inverted indexing" of data – instead of mapping pages to keywords, it maps keywords to pages just like a glossary at the end of any book.

This allows for faster search responses, as it searches through an index, instead of searching through text directly.

Analysis:

An analysis is converting the given text into smaller and precise units for easy the sake of searching.

The text goes through various operations of extracting keywords, removing common words and punctuations, changing words to lower case, etc.

For this purpose, there are multiple built-in analyzers:

1. Standard Analyzer – analyses based on basic grammar, removes stop words like “a”, “an” etc. Also converts in lowercase
2. Simple Analyzer – breaks the text based on no-letter character and converts in lowercase
3. White Space Analyzer – breaks the text based on white spaces

There're more analyzers available for us to use and customize as well.

Query Syntax

Lucene provides a very dynamic and easy to write query syntax.

To search a free text, we'd just use a text String as the query.

To search a text in a particular field, we'd use:

Field Name :text

eg: title:teaCopy

CHAPTER 5

5. IMPLEMENTATION:

5.1. SAMPLE CODE:

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

imports.razor

```
@using System.Net.Http
@using Microsoft.AspNetCore.Authorization
@using Microsoft.AspNetCore.Components.Authorization
@using Microsoft.AspNetCore.Components.Forms
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.AspNetCore.Components.Web.Virtualization
@using Microsoft.JSInterop
@using Examine.Web.Demo
@using Examine.Web.Demo.Shared
```

Examine.test

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Library</OutputType>
    <SccProjectName>
    </SccProjectName>
    <SccLocalPath>
    </SccLocalPath>
    <SccAuxPath>
    </SccAuxPath>
    <SccProvider>
    </SccProvider>
  </PropertyGroup>
  <PropertyGroup>
    <TargetFrameworks>net6.0;</TargetFrameworks>
    <IsPackable>false</IsPackable>
    <GenerateAssemblyInfo>false</GenerateAssemblyInfo>
```

```
</PropertyGroup>
<ItemGroup>
  <Content Include="App_Data\StringTheory.pdf">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
  <None Update="App_Data\PDFStandards.PDF">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="App_Data\TemplateIndex\segments_2">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="App_Data\TemplateIndex\_0.cfs">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="App_Data\umbraco.config">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    <SubType>Designer</SubType>
  </None>
  <None Update="App_Data\TemplateIndex\segments.gen">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <Content Include="App_Data\VS2010CSharp.pdf">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
  <None Update="App_Data\UmbracoContour.pdf">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
</ItemGroup>
<ItemGroup>
  <ProjectReference Include="..\Examine.Core\Examine.Core.csproj" />
  <ProjectReference Include="..\Examine.Host\Examine.csproj" />
  <ProjectReference Include="..\Examine.Lucene\Examine.Lucene.csproj" />
</ItemGroup>
<ItemGroup>
  <Content Include="App_Data\media.xml">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
</ItemGroup>
<ItemGroup>
  <PackageReference Include="Azure.Storage.Blobs" Version="12.13.1" />
  <PackageReference Include="Lucene.Net.Spatial">
    <Version>4.8.0-beta00016</Version>
  </PackageReference>
  <PackageReference Include="Microsoft.Extensions.Hosting" Version="6.0.1" />
  <PackageReference Include="Microsoft.Extensions.Logging" Version="6.0.0" />
  <PackageReference Include="Microsoft.Extensions.Logging.Console" Version="6.0.0" />
  <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.3.2" />
  <PackageReference Include="Moq" Version="4.18.2" />
  <PackageReference Include="NUnit">
    <Version>3.13.3</Version>
  </PackageReference>
```

```
    <PackageReference Include="Nunit3TestAdapter" Version="4.2.1" />
    <PackageReference Include="System.Data.DataSetExtensions" Version="4.5.0" />
    <PackageReference Include="System.Configuration.ConfigurationManager" Version="6.0.1" />
  </ItemGroup>
  <ItemGroup>
    <Compile Remove="DataServices\TestDataService.cs" />
    <Compile Remove="DataServices\TestLogService.cs" />
    <Compile Remove="Search\MultiIndexSearch.cs" />
  </ItemGroup>
</Project>
```

Examine.lucene

```
<?xml version="1.0" encoding="utf-8"?>
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <GeneratePackageOnBuild>false</GeneratePackageOnBuild>
    <Description>A Lucene.Net search and indexing implementation for Examine</Description>
    <PackageTags>examine lucene lucene.net lucenenet search index</PackageTags>
  </PropertyGroup>
  <ItemGroup>
    <None Remove="AspExamineManager.cs.bak" />
  </ItemGroup>
  <ItemGroup>
    <AssemblyAttribute Include="System.Runtime.CompilerServices.InternalsVisibleToAttribute">
      <_Parameter1>Examine.Test</_Parameter1>
    </AssemblyAttribute>
  </ItemGroup>
  <ItemGroup>
    <Content Include="..\..\assets\logo-round-small.png" Link="logo-round-small.png" Pack="true"
PackagePath="" />
  </ItemGroup>
  <ItemGroup>
    <PackageReference Include="Lucene.Net.QueryParser">
      <Version>4.8.0-beta00016</Version>
    </PackageReference>
    <PackageReference Include="Lucene.Net.Replicator" Version="4.8.0-beta00016" />
    <PackageReference Include="System.Threading">
      <Version>4.3.0</Version>
    </PackageReference>
    <PackageReference Include="System.Threading.AccessControl">
      <Version>4.7.0</Version>
    </PackageReference>
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="..\Examine.Core\Examine.Core.csproj" />
  </ItemGroup>

</Project>
```

query.cs

```
using System;
using System.Collections.Generic;

namespace Examine.Search
{
    /// <summary>
    /// Defines the query methods for the fluent search API
    /// </summary>
    public interface IQuery
    {
        /// <summary>
        /// Passes a text string which is preformatted for the underlying search API. Examine will not
modify this
        /// </summary>
        /// <remarks>
        /// This allows a developer to completely bypass and Examine logic and comprise their own
query text which they are passing in.
        /// It means that if the search is too complex to achieve with the fluent API, or too dynamic to
achieve with a static language
        /// the provider can still handle it.
        /// </remarks>
        /// <param name="query">The query.</param>
        /// <returns></returns>
        IBooleanOperation NativeQuery(string query);

        /// <summary>
        /// Creates an inner group query
        /// </summary>
        /// <param name="inner"></param>
        /// <param name="defaultOp">The default operation is OR, generally a grouped query would
have complex inner queries with an OR against another complex group query</param>
        /// <returns></returns>
        IBooleanOperation Group(Func<INestedQuery, INestedBooleanOperation> inner,
BooleanOperation defaultOp = BooleanOperation.Or);

        /// <summary>
        /// Query on the id
        /// </summary>
        /// <param name="id">The id.</param>
        /// <returns></returns>
        IBooleanOperation Id(string id);

        /// <summary>
        /// Query on the specified field for a struct value which will try to be auto converted with the
correct query
```

```
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="fieldName"></param>
/// <param name="fieldValue"></param>
/// <returns></returns>
IBooleanOperation Field<T>(string fieldName, T fieldValue) where T : struct;

/// <summary>
/// Query on the specified field
/// </summary>
/// <param name="fieldName">Name of the field.</param>
/// <param name="fieldValue">The field value.</param>
/// <returns></returns>
IBooleanOperation Field(string fieldName, string fieldValue);

/// <summary>
/// Query on the specified field
/// </summary>
/// <param name="fieldName">Name of the field.</param>
/// <param name="fieldValue">The field value.</param>
/// <returns></returns>
IBooleanOperation Field(string fieldName, IExamineValue fieldValue);

/// <summary>
/// Queries multiple fields with each being an And boolean operation
/// </summary>
/// <param name="fields">The fields.</param>
/// <param name="query">The query.</param>
/// <returns></returns>
IBooleanOperation GroupedAnd(IEnumerable<string> fields, params string[] query);

/// <summary>
/// Queries multiple fields with each being an And boolean operation
/// </summary>
/// <param name="fields">The fields.</param>
/// <param name="query">The query.</param>
/// <returns></returns>
IBooleanOperation GroupedAnd(IEnumerable<string> fields, params IExamineValue[] query);

/// <summary>
/// Queries multiple fields with each being an Or boolean operation
/// </summary>
/// <param name="fields">The fields.</param>
/// <param name="query">The query.</param>
/// <returns></returns>
IBooleanOperation GroupedOr(IEnumerable<string> fields, params string[] query);
```

```
        /// <summary>
        /// Queries multiple fields with each being an Or boolean operation
        /// </summary>
        /// <param name="fields">The fields.</param>
        /// <param name="query">The query.</param>
        /// <returns></returns>
        IBooleanOperation GroupedOr(IEnumerable<string> fields, params IExamineValue[] query);

        /// <summary>
        /// Queries multiple fields with each being an Not boolean operation
        /// </summary>
        /// <param name="fields">The fields.</param>
        /// <param name="query">The query.</param>
        /// <returns></returns>
        IBooleanOperation GroupedNot(IEnumerable<string> fields, params string[] query);

        /// <summary>
        /// Queries multiple fields with each being an Not boolean operation
        /// </summary>
        /// <param name="fields">The fields.</param>
        /// <param name="query">The query.</param>
        /// <returns></returns>
        IBooleanOperation GroupedNot(IEnumerable<string> fields, params IExamineValue[] query);

        /// <summary>
        /// Matches all items
        /// </summary>
        /// <returns></returns>
        IOrdering All();

        /// <summary>
        /// The index will determine the most appropriate way to search given the query and the fields
provided
        /// </summary>
        /// <param name="query"></param>
        /// <param name="fields"></param>
        /// <returns></returns>
        IBooleanOperation ManagedQuery(string query, string[] fields = null);

        /// <summary>
        /// Matches items as defined by the IIndexFieldValueType used for the fields specified.
        /// If a type is not defined for a field name, or the type does not implement
IIndexRangeValueType for the types of min and max, nothing will be added
        /// </summary>
        /// <typeparam name="T"></typeparam>
```

```
        /// <param name="min"></param>
        /// <param name="max"></param>
        /// <param name="fields"></param>
        /// <param name="minInclusive"></param>
        /// <param name="maxInclusive"></param>
        /// <returns></returns>
        IBooleanOperation RangeQuery<T>(string[] fields, T? min, T? max, bool minInclusive = true,
bool maxInclusive = true) where T : struct;
    }
}
```

Lucenequerysearch.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using Examine.Lucene.Indexing;
using Examine.Search;
using Lucene.Net.Analysis;
using Lucene.Net.Search;

namespace Examine.Lucene.Search
{
    /// <summary>
    /// This class is used to query against Lucene.Net
    /// </summary>
    [DebuggerDisplay("Category: {Category}, LuceneQuery: {Query}")]
    public class LuceneSearchQuery : LuceneSearchQueryBase, IQueryExecutor
    {
        private readonly ISearchContext _searchContext;
        private ISet<string> _fieldsToLoad = null;

        public LuceneSearchQuery(
            ISearchContext searchContext,
            string category, Analyzer analyzer, LuceneSearchOptions searchOptions, BooleanOperation
occurance)
            : base(CreateQueryParser(searchContext, analyzer, searchOptions), category, searchOptions,
occurance)
        {
            _searchContext = searchContext;
        }

        private static CustomMultiFieldQueryParser CreateQueryParser(ISearchContext searchContext,
Analyzer analyzer, LuceneSearchOptions searchOptions)
        {
```

```
        var parser = new ExamineMultiFieldQueryParser(searchContext,
LuceneInfo.CurrentVersion, analyzer);

        if (searchOptions != null)
        {
            if (searchOptions.LowercaseExpandedTerms.HasValue)
            {
                parser.LowercaseExpandedTerms = searchOptions.LowercaseExpandedTerms.Value;
            }
            if (searchOptions.AllowLeadingWildcard.HasValue)
            {
                parser.AllowLeadingWildcard = searchOptions.AllowLeadingWildcard.Value;
            }
            if (searchOptions.EnablePositionIncrements.HasValue)
            {
                parser.EnablePositionIncrements = searchOptions.EnablePositionIncrements.Value;
            }
            if (searchOptions.MultiTermRewriteMethod != null)
            {
                parser.MultiTermRewriteMethod = searchOptions.MultiTermRewriteMethod;
            }
            if (searchOptions.FuzzyPrefixLength.HasValue)
            {
                parser.FuzzyPrefixLength = searchOptions.FuzzyPrefixLength.Value;
            }
            if (searchOptions.Locale != null)
            {
                parser.Locale = searchOptions.Locale;
            }
            if (searchOptions.TimeZone != null)
            {
                parser.TimeZone = searchOptions.TimeZone;
            }
            if (searchOptions.PhraseSlop.HasValue)
            {
                parser.PhraseSlop = searchOptions.PhraseSlop.Value;
            }
            if (searchOptions.FuzzyMinSim.HasValue)
            {
                parser.FuzzyMinSim = searchOptions.FuzzyMinSim.Value;
            }
            if (searchOptions.DateResolution.HasValue)
            {
                parser.SetDateResolution(searchOptions.DateResolution.Value);
            }
        }
```

```
            return parser;
        }

        public virtual IBooleanOperation OrderBy(params SortableField[] fields) =>
OrderByInternal(false, fields);

        public virtual IBooleanOperation OrderByDescending(params SortableField[] fields) =>
OrderByInternal(true, fields);

        public override IBooleanOperation Field<T>(string fieldName, T fieldValue)
            => RangeQueryInternal<T>(new[] { fieldName }, fieldValue, fieldValue, true, true,
Occurrence);

        public override IBooleanOperation ManagedQuery(string query, string[] fields = null)
            => ManagedQueryInternal(query, fields, Occurrence);

        public override IBooleanOperation RangeQuery<T>(string[] fields, T? min, T? max, bool
minInclusive = true, bool maxInclusive = true)
            => RangeQueryInternal(fields, min, max, minInclusive, maxInclusive, Occurrence);

        protected override INestedBooleanOperation FieldNested<T>(string fieldName, T fieldValue)
            => RangeQueryInternal<T>(new[] { fieldName }, fieldValue, fieldValue, true, true,
Occurrence);

        protected override INestedBooleanOperation ManagedQueryNested(string query, string[] fields
= null)
            => ManagedQueryInternal(query, fields, Occurrence);

        protected override INestedBooleanOperation RangeQueryNested<T>(string[] fields, T? min, T?
max, bool minInclusive = true, bool maxInclusive = true)
            => RangeQueryInternal(fields, min, max, minInclusive, maxInclusive, Occurrence);

        internal LuceneBooleanOperationBase ManagedQueryInternal(string query, string[] fields,
Occur occurance)
        {
            Query.Add(new LateBoundQuery(() =>
            {
                //if no fields are specified then use all fields
                fields = fields ?? AllFields;

                var types = fields.Select(f => _searchContext.GetFieldValueType(f)).Where(t => t !=
null);

                //Strangely we need an inner and outer query. If we don't do this then the lucene syntax
returned is incorrect
```

```
            //since it doesn't wrap in parenthesis properly. I'm unsure if this is a lucene issue (assume so) since that is what
            //is producing the resulting lucene string syntax. It might not be needed internally within Lucene since it's an object
            //so it might be the ToString() that is the issue.
            var outer = new BooleanQuery();
            var inner = new BooleanQuery();

            foreach (var type in types)
            {
                var q = type.GetQuery(query);

                if (q != null)
                {
                    //CriteriaContext.ManagedQueries.Add(new KeyValuePair<IIndexFieldValueType, Query>(type, q));
                    inner.Add(q, Occur.SHOULD);
                }
            }

            outer.Add(inner, Occur.SHOULD);

            return outer;
        }), occurance);

        return CreateOp();
    }

    internal LuceneBooleanOperationBase RangeQueryInternal<T>(string[] fields, T? min, T? max, bool minInclusive, bool maxInclusive, Occur occurance)
        where T : struct
    {
        Query.Add(new LateBoundQuery(() =>
        {
            //Strangely we need an inner and outer query. If we don't do this then the lucene syntax returned is incorrect
            //since it doesn't wrap in parenthesis properly. I'm unsure if this is a lucene issue (assume so) since that is what
            //is producing the resulting lucene string syntax. It might not be needed internally within Lucene since it's an object
            //so it might be the ToString() that is the issue.
            var outer = new BooleanQuery();
            var inner = new BooleanQuery();

            foreach (var f in fields)
            {
```

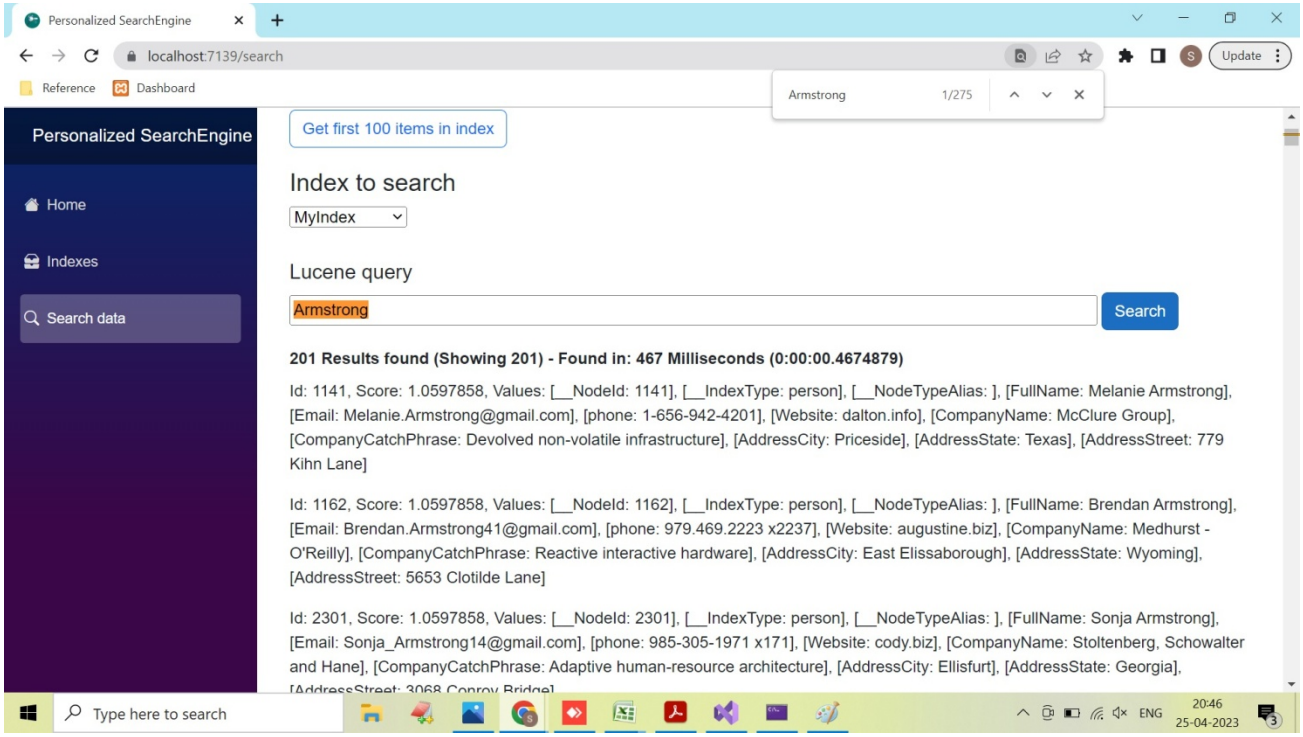
```
                var valueType = _searchContext.GetFieldValueType(f);

                if (valueType is IIndexRangeValueType<T> type)
                {
                    var q = type.GetQuery(min, max, minInclusive, maxInclusive);

                    if (q != null)
                    {
                        //CriteriaContext.FieldQueries.Add(new KeyValuePair<IIndexFieldValueType,
Query>(type, q));
                        inner.Add(q, Occur.SHOULD);
                    }
                }
#if !NETSTANDARD2_0 && !NETSTANDARD2_1
                else if(typeof(T) == typeof(DateOnly) && valueType is
IIndexRangeValueType<DateTime> dateOnlyType)
                {
                    TimeOnly minValueTime = minInclusive ? TimeOnly.MinValue :
TimeOnly.MaxValue;
                    var minValue = min.HasValue ? (min.Value as
DateOnly?)?.ToDateTime(minValueTime) : null;

                    TimeOnly maxValueTime = maxInclusive ? TimeOnly.MaxValue :
TimeOnly.MinValue;
                    var maxValue = max.HasValue ? (max.Value as
DateOnly?)?.ToDateTime(maxValueTime) : null;

                    var q = dateOnlyType.GetQuery(minValue, maxValue, minInclusive, maxInclusive);

                    if (q != null)
                    {
                        inner.Add(q, Occur.SHOULD);
                    }
                }
#endif
                else
                {
                    throw new InvalidOperationException($"Could not perform a range query on the
field {f}, it's value type is {valueType?.GetType()}");
                }
            }

            outer.Add(inner, Occur.SHOULD);

            return outer;
        }), occurance);
```

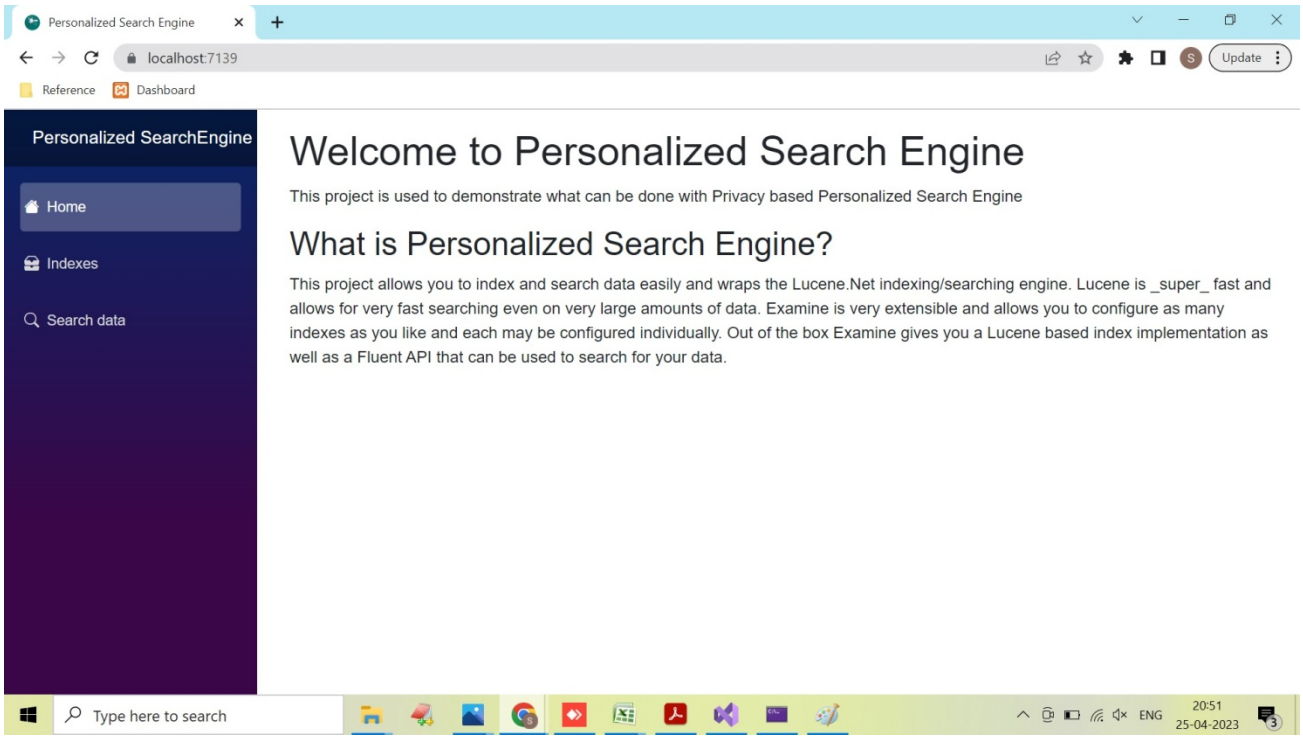
```
            return CreateOp();
        }

        /// <inheritdoc />
        public ISearchResults Execute(QueryOptions options = null) => Search(options);

        /// <summary>
        /// Performs a search with a maximum number of results
        /// </summary>
        private ISearchResults Search(QueryOptions options)
        {
            // capture local
            var query = Query;

            if (!string.IsNullOrEmpty(Category))
            {
                // rebuild the query
                IList<BooleanClause> existingClauses = query.Clauses;

                if (existingClauses.Count == 0)
                {
                    // Nothing to search. This can occur in cases where an analyzer for a field doesn't return
                    // anything since it strips all values.
                    return EmptySearchResults.Instance;
                }

                query = new BooleanQuery
                {
                    // prefix the category field query as a must
                    { GetFieldInternalQuery(ExamineFieldNames.CategoryFieldName, new
ExamineValue(Examineness.Explicit, Category), true), Occur.MUST }
                };

                // add the ones that we're already existing
                foreach (var c in existingClauses)
                {
                    query.Add(c);
                }
            }

            var executor = new LuceneSearchExecutor(options, query, SortFields, _searchContext,
_fieldsToLoad);

            var pagesResults = executor.Execute();
```

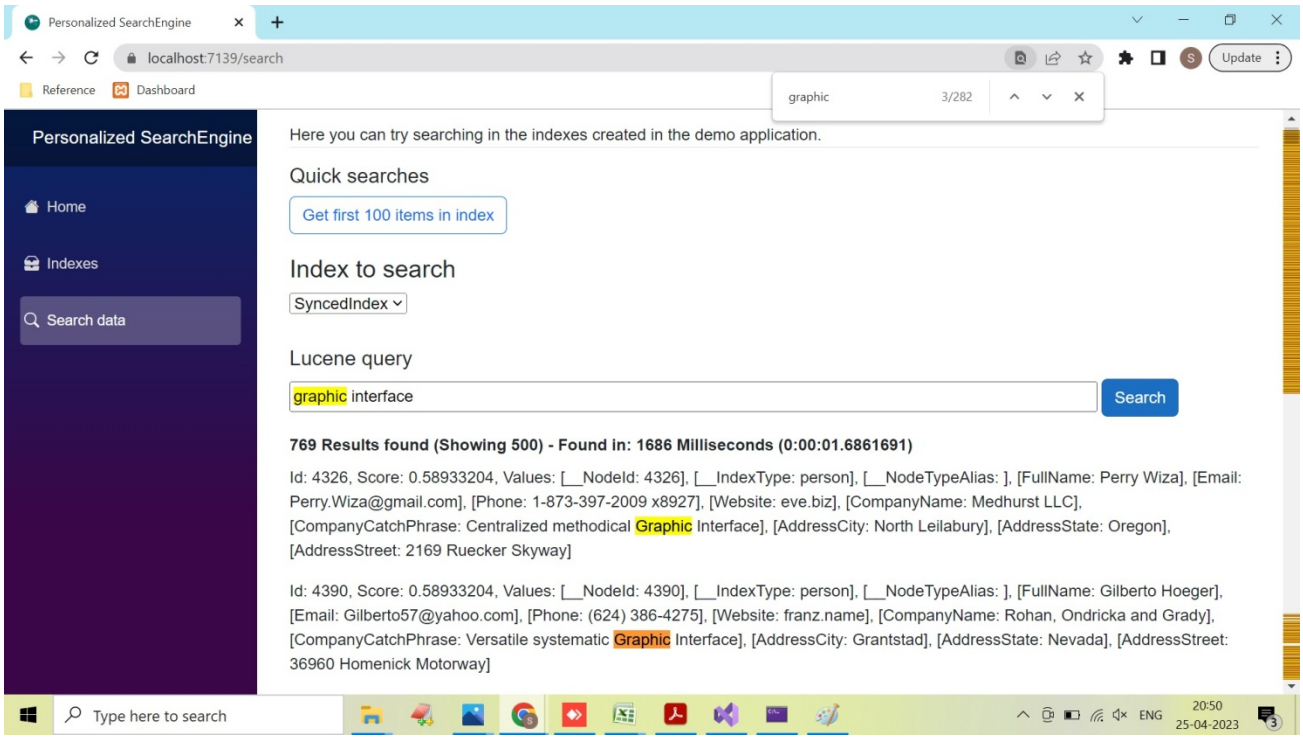
```
            return pagesResults;
        }

        /// <summary>
        /// Internal operation for adding the ordered results
        /// </summary>
        /// <param name="descending">if set to <c>true</c> [descending].</param>
        /// <param name="fields">The field names.</param>
        /// <returns>A new <see cref="IBooleanOperation"/> with the clause appended</returns>
        private LuceneBooleanOperationBase OrderByInternal(bool descending, params
SortableField[] fields)
        {
            if (fields == null) throw new ArgumentNullException(nameof(fields));

            foreach (var f in fields)
            {
                var fieldName = f.FieldName;

                var defaultSort =  SortFieldType.STRING;

                switch (f.SortType)
                {
                    case SortType.Score:
                        defaultSort = SortFieldType.SCORE;
                        break;
                    case SortType.DocumentOrder:
                        defaultSort = SortFieldType.DOC;
                        break;
                    case SortType.String:
                        defaultSort = SortFieldType.STRING;
                        break;
                    case SortType.Int:
                        defaultSort = SortFieldType.INT32;
                        break;
                    case SortType.Float:
                        defaultSort = SortFieldType.SINGLE;
                        break;
                    case SortType.Long:
                        defaultSort = SortFieldType.INT64;
                        break;
                    case SortType.Double:
                        defaultSort = SortFieldType.DOUBLE;
                        break;
                    default:
                        throw new ArgumentOutOfRangeException();
                }
```

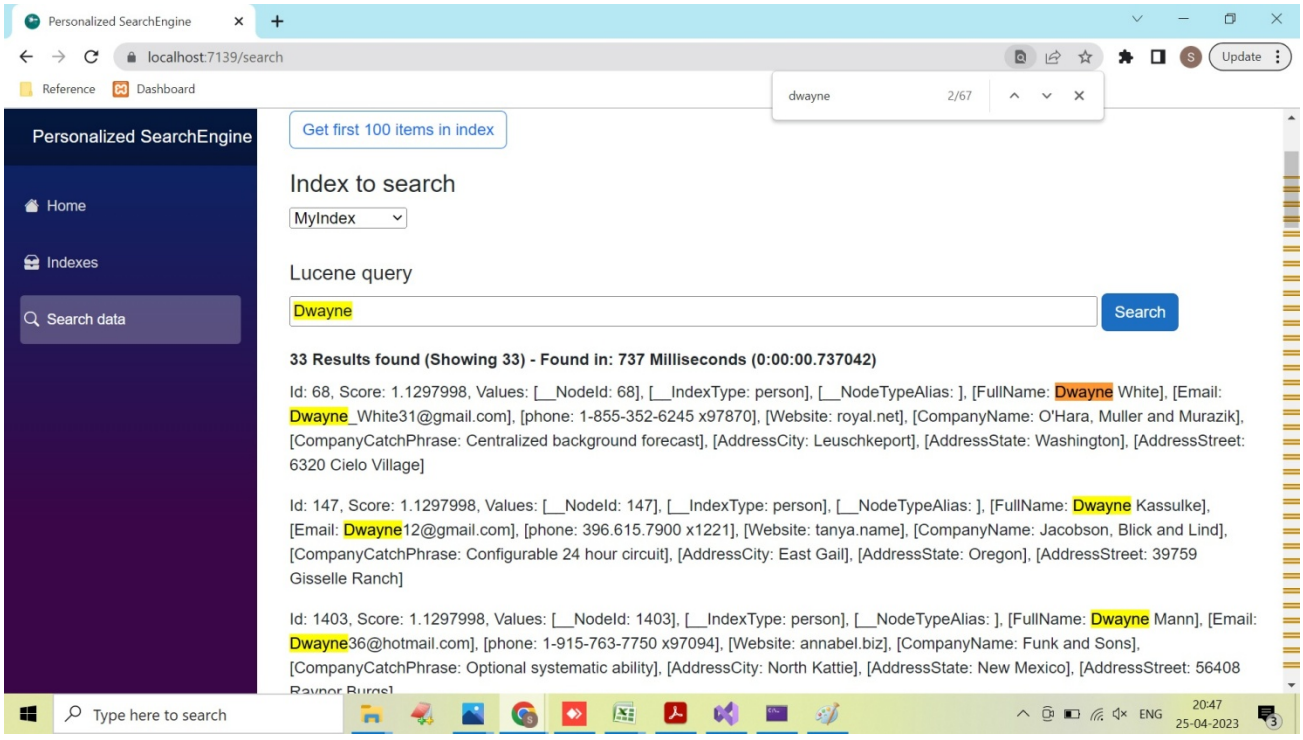
```
            //get the sortable field name if this field type has one
            var valType = _searchContext.GetFieldValueType(fieldName);

            if (valType?.SortableFieldName != null)
                fieldName = valType.SortableFieldName;

            SortFields.Add(new SortField(fieldName, defaultSort, descending));
        }

        return CreateOp();
    }

    internal IBooleanOperation SelectFieldsInternal(ISet<string> loadedFieldNames)
    {
        _fieldsToLoad = loadedFieldNames;
        return CreateOp();
    }

    internal IBooleanOperation SelectFieldInternal(string fieldName)
    {
        _fieldsToLoad = new HashSet<string>(new string[] { fieldName });
        return CreateOp();
    }

    public IBooleanOperation SelectAllFieldsInternal()
    {
        _fieldsToLoad = null;
        return CreateOp();
    }

    protected override LuceneBooleanOperationBase CreateOp() => new
LuceneBooleanOperation(this);

  }
}
```

5.2. SAMPLE OUTPUT SCREENSHOTS:






Personalized SearchEngine
localhost:7139/search
Update
Reference
Dashboard
graphic
3/282
Personalized SearchEngine
Here you can try searching in the indexes created in the demo application.
Home
Indexes
Search data
Quick searches
Get first 100 items in index
Index to search
SyncedIndex
Lucene query
graphic interface
Search
769 Results found (Showing 500) - Found in: 1686 Milliseconds (0:00:01.6861691)
Id: 4326, Score: 0.58933204, Values: [__NodeId: 4326], [__IndexType: person], [__NodeTypeAlias:], [FullName: Perry Wiza], [Email: Perry.Wiza@gmail.com], [Phone: 1-873-397-2009 x8927], [Website: eve.biz], [CompanyName: Medhurst LLC], [CompanyCatchPhrase: Centralized methodical Graphic Interface], [AddressCity: North Leilabury], [AddressState: Oregon], [AddressStreet: 2169 Ruecker Skyway]
Id: 4390, Score: 0.58933204, Values: [__NodeId: 4390], [__IndexType: person], [__NodeTypeAlias:], [FullName: Gilberto Hoeger], [Email: Gilberto57@yahoo.com], [Phone: (624) 386-4275], [Website: franz.name], [CompanyName: Rohan, Ondricka and Grady], [CompanyCatchPhrase: Versatile systematic Graphic Interface], [AddressCity: Grantstad], [AddressState: Nevada], [AddressStreet: 36960 Homenick Motorway]
Type here to search
ENG
20:50
25-04-2023


Personalized SearchEngine
localhost:7139/search
Update
Reference
Dashboard
dwayne
2/67
Personalized SearchEngine
Get first 100 items in index
Home
Indexes
Search data
Index to search
MyIndex
Lucene query
Dwayne
Search
33 Results found (Showing 33) - Found in: 737 Milliseconds (0:00:00.737042)
Id: 68, Score: 1.1297998, Values: [__NodeId: 68], [__IndexType: person], [__NodeTypeAlias:], [FullName: Dwayne White], [Email: Dwayne_White31@gmail.com], [phone: 1-855-352-6245 x97870], [Website: royal.net], [CompanyName: O'Hara, Muller and Murazik], [CompanyCatchPhrase: Centralized background forecast], [AddressCity: Leuschkeport], [AddressState: Washington], [AddressStreet: 6320 Cielo Village]
Id: 147, Score: 1.1297998, Values: [__NodeId: 147], [__IndexType: person], [__NodeTypeAlias:], [FullName: Dwayne Kassulke], [Email: Dwayne12@gmail.com], [phone: 396.615.7900 x1221], [Website: tanya.name], [CompanyName: Jacobson, Blick and Lind], [CompanyCatchPhrase: Configurable 24 hour circuit], [AddressCity: East Gail], [AddressState: Oregon], [AddressStreet: 39759 Gisselle Ranch]
Id: 1403, Score: 1.1297998, Values: [__NodeId: 1403], [__IndexType: person], [__NodeTypeAlias:], [FullName: Dwayne Mann], [Email: Dwayne36@hotmail.com], [phone: 1-915-763-7750 x97094], [Website: annabel.biz], [CompanyName: Funk and Sons], [CompanyCatchPhrase: Optional systematic ability], [AddressCity: North Kattie], [AddressState: New Mexico], [AddressStreet: 56408
Type here to search
ENG
20:47
25-04-2023

CHAPTER 6

6. CONCLUSION AND FUTURE ENHANCEMENTS:

6.1 CONCLUSION:

The effectiveness of search engines on the web is enhanced by personalized web search. PWS employs confidentiality measures to stop the online disclosure of sensitive data. The paper discusses a number of personalization methods. The demand for safety and privacy issues associated with the various personalization strategies were researched. To preserve the confidentiality of users, it also conducted online generalization on profiles of users. The generalization profile was created using a greedy method. In this structure, hierarchical profile building was applied. This may obtain an optimum advancement in future to ensure at most safety, security and privacy in everyone's search.

6.2 FUTURE ENHANCEMENTS:

Personalization tends to be thought of as an important capability for marketing, but we believe it must become the core driver of how companies do marketing. Here's where successful brands need to focus now: Personalization is impossible if marketers don't have the means to understand the needs of high-value customers on an ongoing basis. So top marketers are developing systems that can pool and analyze structured and unstructured data, algorithms that can identify behavioral patterns and customer propensity, and analysis capabilities to feed that information into easy-to use dashboards Making this technological leap forward requires marketing and IT to join forces. A product-management team, with representation from both IT and marketing, should be established to build and refresh the organization's Security and Privacy.

CHAPTER 7

7. REFERENCES

[1]. R. Khan, A. Ahmad, A. O. Alsayed, M. Binsawad, M. A. Islam, and M. Ullah, ‘‘Qupid attack: device gaining knowledge of-based privateness quantification mechanism for PIR protocols in fitness-associated net seek,’’ Sci. software., vol. 2020, Jul. 2020, art. no. 8868686.

[2]. P. Yu, W. U. Ahmad, and H. Wang, ‘‘hide-n-seek: An motive-conscious privacy protection plug-in for personalized web search,’’ in Proc. forty first Int. ACM SIGIR Conf. Res. expand. Inf. Retr., Ann Arbour, MI, united states of America, Jun. 2018, pp. 1333 1336.

[3]. S. B. Mokhtar, G. Berthou, A. Diarra, V. Quema, and A. Shoker, ‘‘RAC: A freerider- resilient, scalable, anonymous communiqué protocol,’’ in Proc. IEEE 33rd Int. Conf. Distrib.Comput. Syst., Philadelphia, PA, usa, Jul. 2013, pp. 520–529.

[4] E. Balsa, C. Troncoso, and C. Diaz, ‘‘OB- PWS: Obfuscation-based private internetsearch,’’ in Proc. IEEE Symp. Secur. privacy, SanFrancisco, CA, united states of America, may additionally 2012, pp. 491–505.

[5] Weber, A. Popescu, and M. Pennacchiotti, ‘‘PLEAD 2012: Politics, elections and facts,’’ in Proc. twenty first ACM Int. Conf. Inf. Knowl. control., 2012, pp. 2768–2769.

[6] H. Corrigan-Gibbs and B. Ford, ‘‘Dissent: accountable nameless group messaging,’’ in Proc. seventeenth ACM Conf. Comput. Commun.Secur. (CCS), Chicago, IL, united states of America, 2010, pp. 340–350.

[7]. Y. Chen, D. Pavlov, and J. F. Canny, ‘‘big- scale behavioural targeting,’’ in Proc. fifteenth ACM SIGKDD Int. Conf. Knowl. Discovery statistics Mining (KDD), 2009, pp. 209–218.

[8]. B. Tancer, click: What tens of millions peopleAre Doing online Why it subjects. Hyperion,2008.

[9]. R. Jones, R. Kumar, B. Pang, and A. Tomkins,‘‘‘I understand what you probably did final summer season’: question logs and consumer privateness,’’ in Proc. 16th ACM Conf. Conf. Inf. Knowl. control. (CIKM), 2007, pp. 909–914.

[10]. Z. Dou, R. tune, and J.-R. Wen, ‘‘A huge-scaleassessment and analysis of customized search techniques,’’ in Proc. sixteenth Int. Conf. international extensive net (WWW), 2007, pp. 581–590.

[11]. F. M. Facca and P. L. Lanzi, ‘‘Mining excitinginformation from weblogs: A survey,’’ statistics Knowl. Eng., vol. fifty three, no. three, pp. 225– 241, Jun. 2005.

[12]. R. Dingledine, N. Mathewson, and P. F. Syverson, ‘‘Tor: The second generation onion router,’’ in Proc. thirteenth USENIX Secur. Symp., San Diego, CA, u.s., Aug. 2004, pp. 303–320.

[13]. M. Eirinaki and M. Vazirgiannis, ‘‘internet mining for web personalization,’’ ACM Trans. internet Technol., vol. three, pp. 1–27, Feb. 2003.