**Enhancing Search Engine User Experience Through Personalization**

*A Project report submitted in partial fulfillment of the requirements*

*For the award of the Degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

By  
Dwarapureddy Manasa

(Regd. No.: 320136410034)  
E. Divij Vignesh

(Regd. No.: 320136410035)  
Chekuri Ashish Varma

(Regd. No.: 320136410024)

Kadimi Harsh

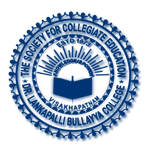
(Regd. No.: 320136410048)

Under the esteemed guidance of

**Mr. Syed Mujib Rahaman**

Associate Professor

Dr. L. Bullayya College of Engineering



**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**Dr. L. BULLAYYA COLLEGE OF ENGINEERING**

(Permanently Affiliated to Andhra University, Visakhapatnam)

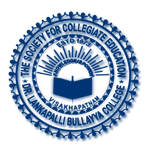
New Resapuvanipalem, Visakhapatnam-530013

Year of submission:2024

**Dr. L. BULLAYYA COLLEGE OF ENGINEERING**

**New Resapuvanipalem, Visakhapatnam-530013**

**Department of Computer Science Engineering**



**Bonafide certificate**

This is to certify that **Ms. D. Manasa, Mr. E. Divij Vignesh, Mr. Ch. Ashish Varma,** and **Mr. K. Harsh** bearing register numbers **320136410034**, **320136410035**, **320136410024**, and **320136410048** students in final yearB.Tech in Computer Science Engineering, have carried out the project work titled ”**Enhancing Search Engine User Experience Through Personalization”** during the academic year **2023-24.**

**Project Supervisor Head of the Department**

**Mr. Syed Mujib Rahaman**  **Dr. D. Madhavi**

Associate Professor Professor

Dr. L. Bullayya College of Engineering Dept. of Computer Science Engineering

# **Abstract**

In the landscape of internet exploration and knowledge-based information retrieval, search engines play a pivotal role. However, the extensive collection and processing of user data raise significant concerns regarding user permission, privacy, and data security. This encompasses the entire search engine environment, from data collection and storage to its utilization by advertisers, potentially leading to unauthorized access and data breaches.

Privacy concerns extend beyond mere data collection, as search engines leverage this information to generate personalized ads in collaboration with third-party advertisers. This often occurs without users' full knowledge or consent, resulting in user discomfort and skepticism.

Additionally, retrieving data from personal knowledge bases, such as databases and documents, poses a challenge. Many companies possess substantial amounts of data related to their policies and client information, making data retrieval cumbersome. Existing data search engines in the market, though numerous, often fail to provide a satisfactory user experience.

Motivated by these challenges, our project aims to enhance user experience in internet searches and data retrieval from personal knowledge bases.

To address privacy concerns, our proposed solution emphasizes protecting users' private search data. The risk of unauthorized access or data breaches poses a significant threat, and safeguarding user data is crucial. Achieving a balance between privacy and user experience is vital, particularly in the context of web searches where search history may be stored and potentially sold to third-party companies.

Improving the user experience in data retrieval engines is imperative, emphasizing the proper understanding of user queries and the retrieval of relevant information. This research endeavors to contribute solutions that prioritize user privacy while enhancing the overall search experience.

### 

### 

### **Acknowledgment**

We would like to express our heartfelt gratitude to **Prof D. Deepak Chowdary, Principal**, Dr. L. Bullayya College of Engineering for giving us the opportunity to do this project. We would like to thank **Dr. D. Madhavi, HOD**, Department of CSE, Dr. L. Bullayya College of Engineering, Visakhapatnam, for her guidance in producing this work. We are deeply indebted to our head of research cluster and the project guide “Privacy-Preserving Network Security in Data Science and Deep Learning” **Mr. Syed Mujib Rahaman**, **Associate Professor**, Department of CSE, Dr. Lankapalli. Bullayya College of Engineering, Visakhapatnam, for guiding us throughout the project despite her busy schedule. Apart from our efforts, the success of this project depends largely on the encouragement of other faculty of CSE, Dr. Lankapalli. Bullayya College of Engineering, Visakhapatnam. We take this opportunity to express our gratitude to the entire faculty who has been instrumental in the successful completion of this project. Also deserving of thanks to our family and friends, for their support for their confidence in our achievements.

**D. Manasa  
E. Divij Vignesh**

**Ch. Ashish Varma  
Kadimi Harsh**

### 

### 

### 

### **Declaration**

This is to declare that the Project work entitled “**Enhancing Search Engine User Experience Through Personalization**” is a bonafide work done by us under the research cluster group “Privacy-Preserving Network Security in Data Science and Deep Learning” with the esteemed guidance of **Mr. Syed Mujib Rahaman, Associate Professor**, Department of CSE, Dr. L. Bullayya College of Engineering. This project report is being submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering during the academic year 2023-2024. This project possesses originality as it is not extracted from any source and it has not been submitted to any other institutions and universities.

(D. Manasa)

Reg.No: 320136410034

(E. Divij Vignesh)

Reg.No:320136410035

(Ch. Ashish Varma)

Reg.No:320136410024

(Kadimi Harsh)

Reg.No:320136410048

Visakhapatnam

Date:

**Title Page No**

[**Abstract**](#_im8z9rdt74as) **3**

[Acknowledgment](#_cl8jqdgypczg) 4

[Declaration](#_47tmlroumat4) 5

[List of Tables](#_j7xwr3ne4qxa) 8

[List of Figures](#_6oqdfcw3mp0n) 8

[**1. Introduction**](#_bvc9kdu29wya) **9**

[1.1 Background and motivation](#_c5xnaab7oya0)9

[1.2 Problem Statement](#_kz1yy9iqir39) 9

[**2. Requirement Elicitation and Analysis**](#_dt4sn55ivaqz) **11**

[2.1 Existing System](#_jyczq9i1okwd)11

[2.2 Proposed System](#_whld5ek3599g) 11

[2.3 System Requirement](#_e0x7ilb97m8) 12

[2.3.1 Functional Requirements](#_nppq2j56ydq5) 13

[2.3.2 Non-Functional Requirements](#_fguaj8655j39) 13

[2.3.3 Hardware Requirements](#_rpp9bzbnqpiz) 14

[2.3.4 Software Requirements](#_bzhi2h8x8mo7) 14

[2.4 Feasibility Study](#_8e9lh0danukg)15

[**3. System Design**](#_nc3zlrqvlip1) **17**

[3.1 Object-Oriented Analysis & Design](#_7i5v3lq60z0s) 18

[Table: UML Symbols](#_313anu4bageb) 19

[3.1.1 Scenarios](#_940ganbudjsa) 20

[3.1.2 Use Case Diagrams](#_e7m9mqkyxyu5) 21

[3.1.3 Activity Diagrams](#_z9l0v6cp2avv) 23

[3.1.4 Sequence Diagrams](#_z8yzklf78bdk) 24

[3.1.5 Class Diagram](#_vx9qh8u5ykev) 25

[**4. Implementation**](#_13owzybysy48) **26**

[4.1 Software Environment](#_l3vfuqjths40) 26

[4.2 Neural Network Architecture](#_wdzc7mrsvzhr) 27

[4.3 Software Technologies](#_fwvt5u6veoqz) 29

[**5. Testing**](#_cphspotnh622) **30**

[5.1 Testing Methodologies](#_223b8tjdwkdp) 30

[5.1.1 Black Box vs. White Box Testing](#_p1re766sznhe) 30

[5.1.2 Unit Testing](#_5x2cp324j6uq) 30

[5.1.3 System Testing](#_if0emimaafv0) 30

[5.1.4 Acceptance Testing](#_s452l1kegyx9) 31

[5.1.5 Quality Assurance](#_g1039ni3y6c6) 31

[**6. Conclusion**](#_ddcacvmr3a46) **34**

[**References:**](#_wz4yppydbnv8) **35**

[**Appendix**](#_sbpc1ppgkqwk) **36**

[Input/Output Screens](#_tdw2ndn3esvo) 36

[Sample Code:](#_igo0q5a1dztf) 38

### 

### 

### 

### **List of Tables**

**S. No Table Name Page No.**

1. SRS Table 11

2. Symbol Table 20

3. Test Case 27

### **List of Figures**

| **S. No** | **Table Name** | **Page No.** |
| --- | --- | --- |
| 1. | Use case Diagram | 20 |
| 2. | Activity Diagram | 21 |
| 3. | Sequence Diagram for analysis | 22 |
| 4. | Class Diagram | 23 |
| 5. | Transformers Architecture | 26 |
| 6. | Representation of different Databases | 34 |
| 7. | Internet Search Modifier Interface | 35 |
| 8. | Knowledge Retriever Interface | 35 |
| 9. | Search result query along with the output | 35 |
| 10. | Knowledge Retriever input and output | 36 |

# **1. Introduction**

## **1.1 Background and motivation:**

## Search engines are widely used as essential tools for internet exploration or information retrieval from knowledge bases. Depending on the vast gathering and processing of user data, however, creates major concerns regarding user permission, privacy, and data security.

This includes the whole search engine environment, from user data collecting and storage to its usage by advertisers and the possibility of unauthorized access or data breaches.

Privacy issues go beyond simply collecting data. Search engines use this information to generate personalized ads in partnership with third-party advertisers. Despite being profitable, this method frequently happens without full knowledge or agreement from users, which causes discomfort and skepticism among users.

Apart from this, retrieving data from personal knowledge bases( like databases and documents) has been a challenge. Companies have large amounts of data related to their policies, client information, etc. which makes data retrieval quite a hassle.

Searching the required information from this large data can be done with the help of a Data Search Engine. Although there are a lot of Data search engines in the market, they fail to provide proper user experience.

These problems have motivated us to pursue the idea of enhancing user experience in internet search experience and data retrieval from personal knowledge bases.

## **1.2 Problem Statement:**

* Query Understanding and Reformulation:

Develop new algorithms that understand user queries with the help of sophisticated machine learning models and provide query suggestions and reformulation to provide accurate search results.

* Semantic Search Enhancement:

There is a need for developing new algorithms with the help of machine learning models to improve the Semantic Search capabilities of the Data Search Engine, allowing users to find relevant information in knowledge bases based on the meaning and context of the query.

* Personalized Search and Recommendation:

With the help of users' search history, personalized search recommendations from knowledge bases, taking into account user preferences and behavior to provide more relevant search results.

Protecting the private search data of the users is important as no user will be willing to leak his/her data for the sake of improving their user experience. The users’ sensitive information might be breached or could have unauthorized access, which is a significant risk. Concerns over the security of personal data carried on search engine servers continue to rise among privacy-conscious users.

Achieving both privacy and user experience should be prioritized by any service provider. Especially in web searches, the search history is stored by the service providers which might be sold to other 3rd party companies. To protect the user data and improve the search experience we are suggesting the following solution.

Improving the user experience in data retrieval engines is essential. A good data retrieval engine should be able to properly understand the user query and retrieve the appropriate information.

# 

# 

# 

# **2. Requirement Elicitation and Analysis**

## **2.1 Existing System:**

## The current landscape of search engines is dominated by giants such as Google, Bing, and Yahoo. While these platforms have revolutionized information retrieval, the fundamental approach to search remains fairly uniform. These search engines operate on sophisticated algorithms that analyze keywords, links, and various other factors to generate search results.

Traditional search engines typically employ keyword-matching algorithms to retrieve results based on the exact or near-exact match of user-entered keywords. While this method is straightforward, it often leads to irrelevant and overwhelming results, particularly when dealing with complex or ambiguous queries.

Some search engines offer auto-complete suggestions, these are generally static and not tailored to the user's evolving preferences or recent search patterns.

Existing systems often collect vast amounts of user data for the purpose of improving search accuracy and providing targeted advertisements. This raises privacy concerns among users who may be uncomfortable with the extent of data collection and usage.

## **2.2 Proposed System:**

To achieve both the user experience and data privacy, we propose the following solution:

* Create a model that will improve users' input query and provide personalized search suggestions.
* Creating a website through which users will be able to perform both knowledge retrieval and modify web search queries.

The model which will be created is personalized i.e. each individual user will have their own model weights which will make the model output vary from person to person. These weights are modified based on the user's search queries; this personalized weight update will ensure that the user experience is improved and upheld.

When a user searches for a particular query, the searched query data will be stored temporarily in the backend for training the model and improving the model weights for that particular user. Once the training and updation of the weights has been done, the search data will be deleted and only the model weights will be stored in the backend.

Deleting the search query will ensure that the search data is not used for any other purpose other than training the model. Storing the modified weights cannot be a privacy threat as it is impossible to reverse engineer the model weights to get back the original data.

This approach will ensure that both the data privacy and user experience are satisfied. Another advantage of this approach is that each user will have their own search query modification model and this will provide a promising user search experience.

## **2.3 System Requirement:**

### **2.3.1 Functional Requirements:**

* The Model should accurately analyze user search history, demographics, and browsing patterns.
* It should provide personalized suggestions and improve query prompts based on user context.
* It should deliver more relevant and accurate search results.
* User authentication and search history tracking.
* Database for storing and retrieving user data.

| **Requirement ID** | **Requirement** | **Priority** | **Type** | **Source** | **Description** |
| --- | --- | --- | --- | --- | --- |
| SRS-1 | Search Query Submission | 1 | Functional | User Story | The user should be able to enter a search query into the search bar. |
| SRS-2 | Search Query Processing | 1 | Functional | User Story | The search engine system should analyze the user's search query, including its keywords, intent, and context. |
| SRS-3 | Relevant Search Results Retrieval | 1 | Functional | User Story | The search engine system should retrieve relevant search results from its database, considering the user's search query and context. |
| SRS-4 | User Context Analysis | 2 | Functional | User Story | The search engine system should analyze the user's context, including their search history, demographics, and browsing patterns, to enhance personalization. |
| SRS-5 | Personalized Suggestions Generation | 2 | Functional | User Story | The search engine system should generate personalized suggestions based on the user's context and the retrieved search results. |
| SRS-6 | Personalized Search Results Display | 1 | Functional | User Story | The search engine system should combine relevant search results with personalized suggestions and present them to the user in an organized and user-friendly manner. |

| **Requirement ID** | **Requirement** | **Priority** | **Type** | **Source** | **Description** |
| --- | --- | --- | --- | --- | --- |
| SRS-7 | Performance | 3 | Non-Functional | Stakeholder Requirement | The search engine system should provide a responsive and efficient search experience, with minimal latency and acceptable load times. |
| SRS-8 | Accuracy | 3 | Non-Functional | Stakeholder Requirement | The search engine system should provide accurate and relevant search results, considering the user's intent and context. |
| SRS-9 | Ease of Use | 3 | Non-Functional | Stakeholder Requirement | The search engine system should provide an intuitive and easy-to-use interface that is accessible to users of varying technical expertise. |

Table 1: SRS Table

### **2.3.2 Non-Functional Requirements:**

* The Model should be scalable to handle large volumes of user data and search queries.
* It should be responsive and provide real-time suggestions.
* Maintain user privacy and data security.
* Secure storage and transmission of user data.
* Users of any level of expertise should find this model to be simple to use.
* Users should always be able to access the model and it should be reliable.
* This model should be accessible to users with disabilities.

### **2.3.3 Hardware Requirements:**

* **CPU**: A multi-core processor with decent clock speed is recommended for efficient code execution and handling of background tasks.
* **RAM**: At least 16GB of RAM is recommended to ensure smooth operation, especially while working with large datasets or complex models.
* **Storage**: Sufficient storage space (SSD preferred for faster data access) is needed to accommodate the project's code, datasets, and any pre-trained models.
* **Graphics Processing Unit (GPU):** This is the crucial component for the project, as it allows for efficient training and running of LLMs.
* **GPU Memory**: The required GPU memory (VRAM) depends heavily on the specific LLM models used in the project.

For smaller LLMs, a GPU with 8GB of VRAM might be sufficient.

For larger LLMs, a GPU with 16GB or even 32GB of VRAM might be necessary.

**Additional Considerations:**

* **Internet connection**: A stable internet connection is necessary for downloading libraries and pre-trained models.
* **Power supply**: Ensure your system has a sufficient power supply unit (PSU) to handle the additional power draw of the GPU.

### **2.3.4 Software Requirements:**

* **Operating System**: While not strictly required, a Linux-based operating system is highly recommended due to its open-source nature.
* **Python**: Version 3.6 or later is required to ensure compatibility with the utilized libraries and frameworks.
* **Code Editor or IDE**: Options like Visual Studio Code, PyCharm, and Spyder provide user-friendly interfaces for writing, editing, and debugging Python code.
* **Package Manager**: pip is the standard package manager for Python and is used to install necessary libraries and frameworks.
* **PyTorch**: Version 1.7 or later is recommended for deep learning functionalities, especially if neural network models are used for personalization or knowledge retrieval tasks.
* **ChromaDB**: Version 0.5 or later serves as the vector database for storing and retrieving vector representations of text data.
* **Langchain**: Version 0.9 or later (optional) provides functionalities for working with large language models, potentially used for knowledge retrieval or other relevant tasks.

**Additional Software:**

* **Git version control system** (optional): Facilitates code versioning and collaboration, especially if working with a team or for personal project management.
* **GPU drivers**: Ensure you have the latest drivers installed for your specific GPU model to guarantee optimal performance.

## **2.4 Feasibility Study:**

## The feasibility study involves evaluating the technical, economic, and operational aspects of the project.

1. **Technical Feasibility:**

The proposed system is technically feasible due to the availability of sophisticated machine learning and natural language processing (NLP) technologies. These technologies can effectively analyze large volumes of user data to extract meaningful insights and personalize search experiences. The technical aspects are feasible with the right expertise.

Our project is made with the use of the latest Machine learning and Deep learning tools which made our software more feasible to build and deploy on large scales.

1. **Economic Feasibility:**

The economic Feasibility depends on the development costs and expected revenue. The potential revenue from improved user experience and increased user engagement could outweigh these costs. The economic feasibility is favorable, given the potential long-term gains.

With the latest AI developments happening, as of 2024, the compute required to run these heavy and sophisticated deep learning models is pretty much less. Thanks to the companies building new specialized hardware for training machine learning models and providing them to developers at cheaper prices.

1. **Operational Feasibility:**

The Operational Feasibility of this system depends on its integration into existing search engine infrastructure, user acceptance, and continuous improvement. The system should be seamlessly integrated into search engines to provide a better user experience.

This project is created in such a way that it requires minimum effort to modify and integrate it into different applications. Since compute required is becoming cheaper maintaining and operating this tool is also easier for developers.

# 

# 

# 

# 

# 

# 

# 

# 

# **3. System Design**

## **3.1 Object-Oriented Analysis & Design:**

Object-Oriented Analysis & Design (OOAD) breaks down complex systems into real-world objects and their interactions. This approach, using concepts like classes and inheritance, helps create modular, maintainable, and scalable software. It involves analyzing the problem, modeling it with objects, designing it with classes, and finally implementing it using an object-oriented programming language.

While requiring an understanding of specific concepts and potentially leading to over-engineering in simpler cases, OOAD offers significant benefits in terms of code reusability, readability, and maintainability, making it valuable for complex systems and projects where frequent changes and extensions are expected.

While OOAD provides a powerful framework, it's important to consider its limitations. The initial learning curve can be steep, requiring developers to grasp object-oriented concepts like inheritance and encapsulation. Additionally, applying OOAD to smaller projects might be an unnecessary burden, potentially leading to over-engineering and hindering development speed.

However, for complex systems involving numerous interacting components and a need for frequent updates, OOAD shines. Its emphasis on modularity promotes code reusability, making it easier to adapt and extend the software over time.

This, combined with improved code readability due to its real-world entity representation, makes OOAD a valuable tool for developing and maintaining complex, scalable, and sustainable software solutions.

Beyond the technical aspects, OOAD also fosters a specific way of thinking about software development. By analyzing problems in terms of objects and their interactions, it encourages developers to view the system from a user-centric perspective.

This focus on real-world entities can lead to more intuitive and user-friendly software since the design directly reflects the users' mental models of the problem domain. Additionally, the modular nature of OOAD facilitates collaborative development, as different developers can work on distinct objects and their functionalities independently, leading to faster development cycles and better communication within the team.

### **Table: UML Symbols**

## 

| **S. no.** | **Symbol Name** | **Symbol** | **Description** |
| --- | --- | --- | --- |
| **1.** | Class |  | Classes represent a  collection of similar  entities grouped together. |
| **2.** | Association |  | Association represents a  static relation between  classes. |
| **3.** | Aggregation |  | It aggregates several  classes into a single class. |
| **4.** | Composition |  | Composition is a special  type of aggregation that  denotes strong ownership  between classes. |
| **5.** | Actor |  | Actor is the user of the  system that reacts with the  system. |
| **6.** | Use case |  | A use case is an interaction  between the system and the  external environment |
| 7. | Relation(uses) |  | It is used for additional  purposes of  communication |
| 8. | Communication |  | It is the communication  between use cases. |
| 10. | Initial State |  | It represents the initial  state of the object. |
| 11. | Final State |  | It represents the final state  of the object. |
| 12. | Control Flow |  | It represents the decision-making process for objects |
| 13. | Decision box |  | It represents the decision-making process from a constraint. |
| 14. | Data process/state |  | A circle in a DFD  represents a state or  process that has been  triggered due to some other  event or action. |
| 15. | External entity |  | It represents external  entities such as keyboards,  sensors, etc which are used  in the system. |
| 16. | Transition |  | It represents any communication that occurs between processes |
| 17. | Object lifeline |  | represents the vertical  dimension that the object  communicates. |
| 18. | Message |  | It represents messages  exchanged. |

Table 2: Symbol Table

### **3.1.1 Scenarios:**

**Basic Search Scenario:**

Description: A user enters a search query into the data retrieval engine to find information about a specific topic. The engine processes the query and retrieves relevant data from its knowledge base, presenting the results to the user.

**Personalized Search Scenario:**

Description: A registered user logs into the data retrieval engine. The engine, leveraging the user's search history and preferences, tailors the search results to provide personalized and relevant information based on the user's past interactions.

**Semantic Search Enhancement Scenario:**

Description: The data retrieval engine incorporates advanced semantic search algorithms. When a user submits a query, the engine interprets the meaning and context of the query, providing more accurate and contextually relevant results.

**Real-Time Data Retrieval Scenario:**

Description: A user requires up-to-date information and performs a real-time search using the data retrieval engine. The engine continuously updates its knowledge base and provides the most recent and relevant data in response to the user's query.

**Interactive Search Scenario:**

Description: A user engages in an interactive search session with the data retrieval engine. The engine provides suggestions, allows the user to refine the query, and dynamically updates the search results based on the user's interactions.

**Secure Data Retrieval Scenario:**

Description: Emphasizing data security, a user performs a search while ensuring that sensitive information remains confidential. The data retrieval engine employs encryption and access control measures to protect the privacy of the user's search data.

### 

### **3.1.2 Use Case Diagram:**

A use case diagram is a visual representation of a system's functionality from the perspective of its users. It depicts the interactions between users (called actors) and the system through use cases, which are specific goals or tasks that actors can perform using the system. These diagrams are part of the Unified Modeling Language (UML) and focus on what the system does rather than how it does it.

Here's a breakdown of the key elements:

**Actors:** Represent external entities that interact with the system, such as users, other systems, or external devices.

**Use Cases:** Ellipses depicting functionalities or goals that actors can achieve through interactions with the system.

**Relationships:** Arrows connecting actors and use cases, showing how actors initiate and participate in use cases.

### 

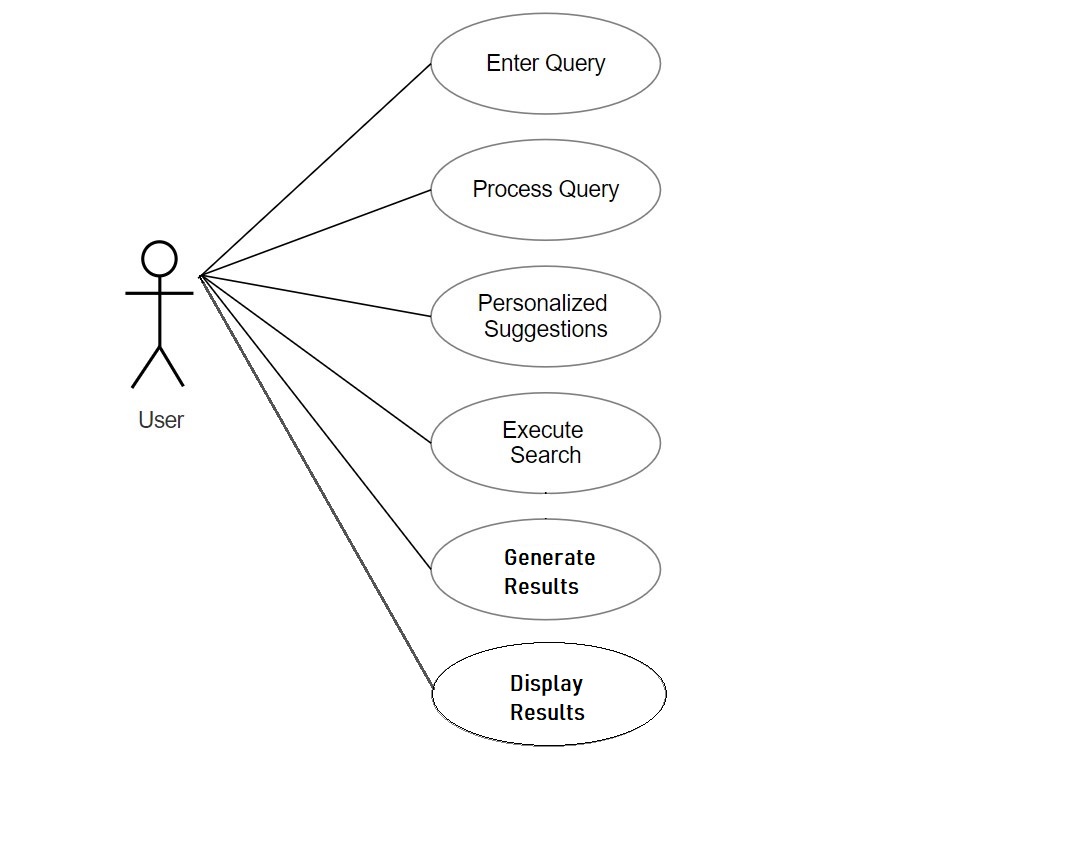
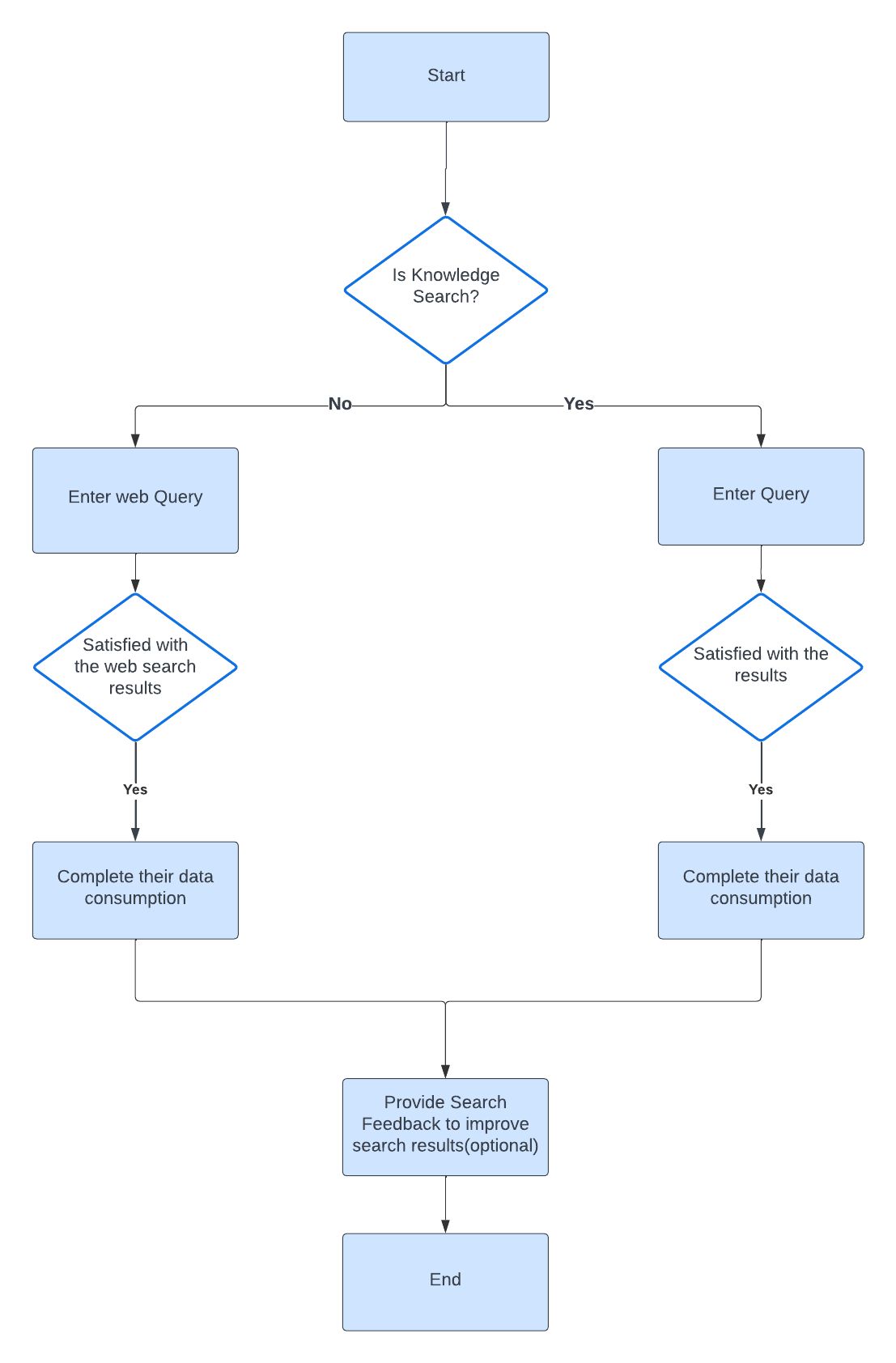


Figure 1: Use Case Diagram

### 

### **3.1.3 Activity Diagram:**

An activity diagram is a visual representation of a workflow within a system. It depicts the sequential steps involved in a process, along with decision points, branching paths, and potential iterations. Similar to a flowchart, it uses boxes to represent activities and arrows to show the flow of control. Activity diagrams are particularly useful for modeling business processes, user interactions with a system, or the internal workings of an algorithm.

  
Figure 2: Activity Diagram

### **3.1.4 Sequence Diagram:**

A sequence diagram is a visual representation of how objects interact with each other in a specific scenario. It depicts the sequence of messages exchanged between objects in chronological order, along with the lifelines of those objects. This allows you to see the flow of communication and how objects collaborate to achieve a particular task. Sequence diagrams are particularly useful for understanding complex interactions and identifying potential bottlenecks or errors in the communication process.

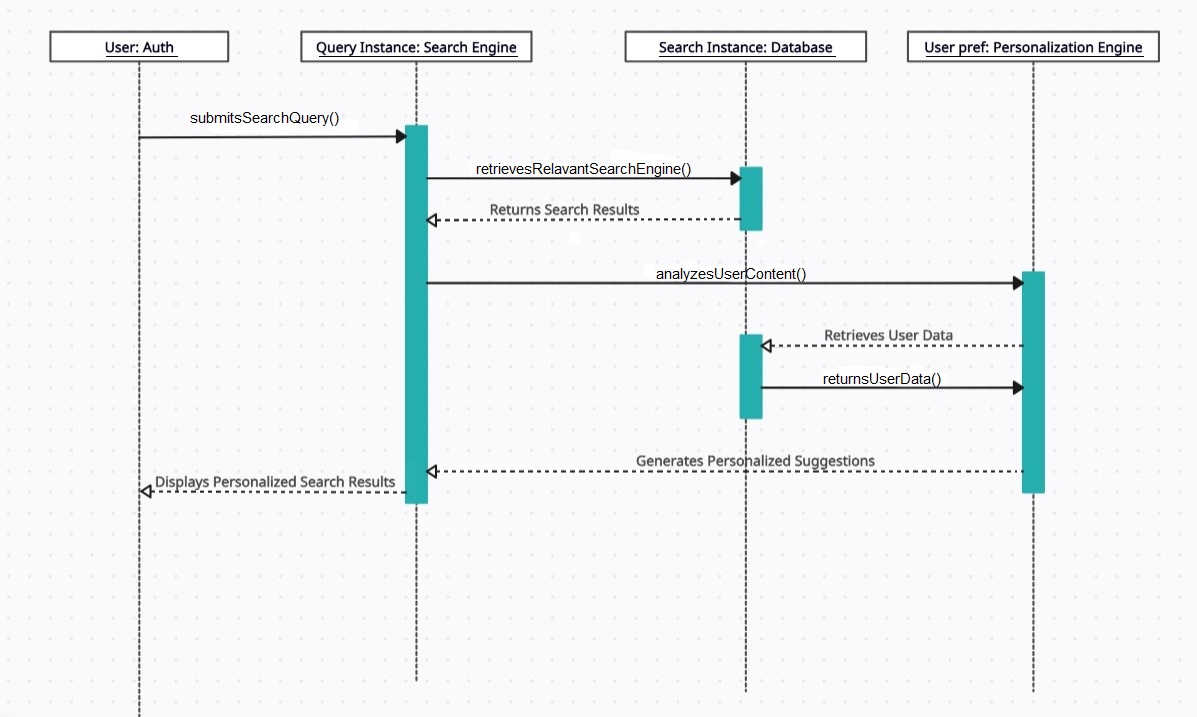


Figure 3: Sequence Diagram

### **3.1.5 Class Diagram:**

A class diagram is a blueprint in software engineering. It visually represents the building blocks (classes) of a system, their internal properties (attributes), the actions they can perform (methods), and the relationships between them. Imagine it as a map outlining the different cities (classes) within a country (software system), what landmarks they have (attributes), and the roads connecting them (relationships). These diagrams are particularly helpful for understanding object-oriented systems and are used during system design and documentation.

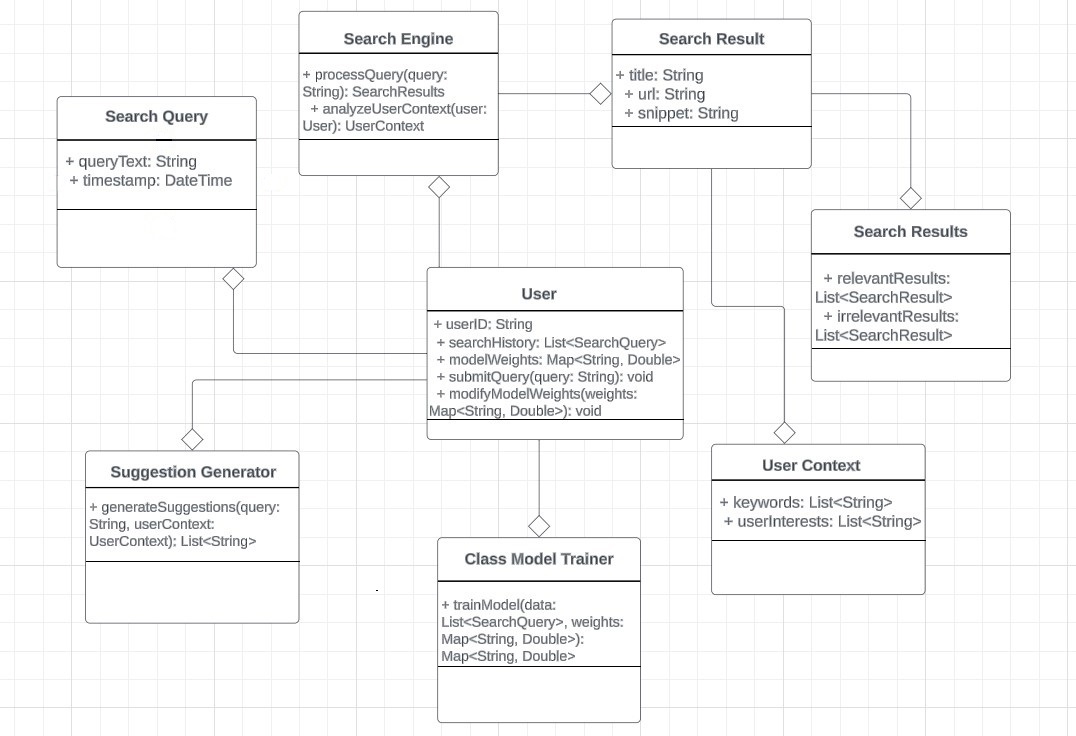


Figure 4. Class Diagram

## 

# **4. Implementation**

## **4.1 Software Environment:**

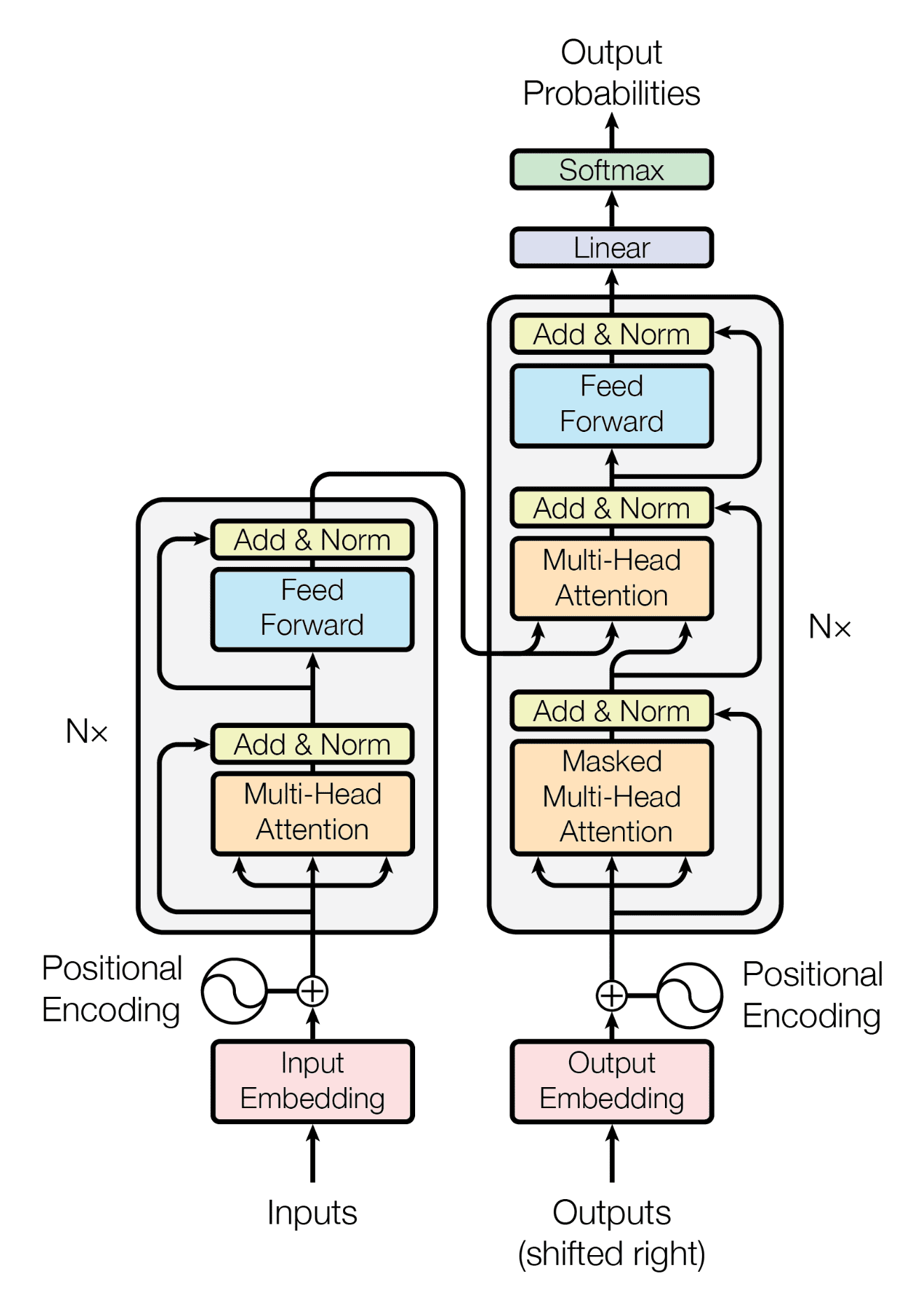
The project's software environment was primarily focused on a Python-based development environment within the local system. This suggests the following configuration:

* **Operating System**: While not explicitly mentioned, a Linux-based operating system is commonly preferred for Python development and machine learning due to its open-source nature, robust command-line interface, and extensive support for scientific computing tools.
* **Development Tools**: Essential Python development tools like code editors or IDEs (Integrated Development Environments) were likely used for writing, editing, and debugging Python code.
* **Libraries and Frameworks**: This project utilized libraries such as PyTorch and Langchain within the Python environment. These would be installed using package managers like pip.
* **Hardware**: The project's success hinged on the use of a system equipped with a Graphics Processing Unit (GPU). GPUs are crucial for efficiently running large language models (LLMs) due to their parallel processing capabilities. The specific GPU model and available memory would depend on the computational demands of the LLMs used.

## **4.2 Neural Network Architecture:**

**Transformers Neural Network:**

The Transformer neural network architecture, introduced by Vaswani et al. in 2017, revolutionized natural language processing and other sequential data tasks. Unlike traditional neural networks, Transformers rely on self-attention mechanisms to capture relationships between different words in a sequence simultaneously. The architecture consists of an encoder-decoder framework, each comprising multiple layers. The self-attention mechanism allows the model to weigh the importance of different words dynamically, enabling effective processing of long-range dependencies. Positional encoding is incorporated to provide information about the order of words. Transformers have demonstrated remarkable success in various applications, including machine translation, text generation, and language understanding, and have become a foundational architecture in modern deep learning.

 Figure 5. Transformers Architecture

**Vector Database Architecture:**

Vector databases are a specialized type of database designed to store and efficiently retrieve vector embeddings. These embeddings are numerical representations of data points, such as text documents, images, or user profiles. Unlike traditional relational databases that store data in tables with rows and columns, vector databases store data as high-dimensional vectors, allowing for similarity searches and other complex operations.

Vector databases are particularly useful for applications that require fast and accurate similarity searches, such as recommender systems, image retrieval, and personalized search engines. In the context of this project, a vector database was likely used to store the vector representations of search queries and user profiles, enabling the system to identify similar queries and personalize search results based on individual user preferences.

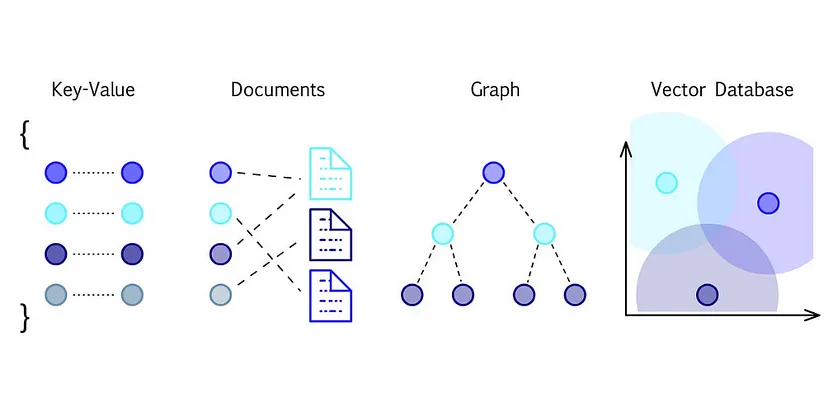


Figure 6. Representation of different Databases

## **4.3 Software Technologies:**

* **Python**: As the primary programming language, Python provides a versatile and well-established platform for building the core functionalities of the system. Its extensive libraries and frameworks facilitated the efficient development and integration of various components.
* **PyTorch**: This deep learning framework played a crucial role in enabling the creation and manipulation of vector representations. PyTorch's capabilities in handling tensors and neural networks were likely leveraged to process user data and extract meaningful vector representations that captured their search preferences and intent.
* **ChromaDB**: This vector database served as the foundation for storing and managing the generated vector representations. ChromaDB's efficient querying capabilities were likely exploited to retrieve relevant search results based on the user's personalized vector and the similarity it shares with other data points within the database.
* **Langchain**: This framework, specifically designed for building large language models (LLMs) and AI-powered applications, provided valuable tools and functionalities for managing and interacting with LLMs and is used for knowledge retrieval or other relevant tasks within the project

# **5. Testing**

In the realm of software development, ensuring a product functions flawlessly and meets user expectations is paramount. This is where testing methodologies come into play. They serve as a structured approach to identify errors, assess functionalities, and refine the overall quality of a software system. This in-depth exploration delves into various testing methodologies, along with their relevance to your final year project on enhancing user search experience.

## **5.1 Testing Methodologies:**

**Black Box Testing**:

This methodology treats the software as a black box, focusing on external functionalities and user interactions. Testers, without knowledge of the internal code, design test cases based on requirements and specifications. It simulates real-world usage patterns to identify issues like usability, compatibility, and adherence to functional requirements.

**White Box Testing**:

In contrast, white box testing, often referred to as glass box testing, delves into the internal workings of the software. Testers with a thorough understanding of the code structure design test cases to ensure proper code flow, logic implementation, and adherence to coding standards. This methodology is particularly effective in uncovering logic errors and code defects.

### **5.1.2 Unit Testing:**

Unit testing forms the foundation of a robust testing strategy. It involves isolating individual units of code (functions, modules) and testing them independently. This granular approach allows for early detection and rectification of errors within smaller code segments, preventing them from cascading into larger functionalities. Unit testing is often automated, enabling efficient and repeatable testing throughout the development lifecycle.

### **5.1.3 System Testing:**

Once individual units are validated, system testing brings them together. Here, the focus shifts to ensuring seamless integration and interaction between various software components. Testers verify if the entire system functions as intended, fulfilling the specified requirements. System testing encompasses functionalities like data exchange, security protocols, and overall system behavior.

### **5.1.4 Acceptance Testing:**

Acceptance testing serves as the final checkpoint before deploying the software to users. It can involve both black-box and white-box testing techniques, with a significant emphasis on user acceptance testing (UAT). Here, actual users or designated representatives evaluate the software against real-world scenarios and provide feedback on its usability, functionality, and suitability for their needs.

### **5.1.5 Quality Assurance:**

Quality Assurance (QA) encompasses a broader set of practices that ensure the overall quality of a software product. It goes beyond just testing and incorporates activities like requirement management, defect tracking, risk assessment, and process improvement. A robust QA framework establishes a culture of quality throughout the development lifecycle, fostering continuous improvement and adherence to best practices.

**Enhancing User Search with Confidence:**

While our project focuses on user search experience using personalization and knowledge retrieval, these testing methodologies can significantly enhance its success.

**Black Box Testing**: Simulate user interactions with your search system to identify usability issues, navigation problems, and relevance of search results.

**Unit Testing**: Validate the core functionalities of your personalization engine and knowledge retrieval algorithms. Ensure they operate as intended on individual data elements.

**System Testing**: Test the integration of your search system with other system components, verifying seamless data exchange and information retrieval.

By employing a combination of these testing methodologies, you can ensure your user search experience delivers personalized and accurate results, fostering user satisfaction and project success. Remember, thorough testing is an investment that pays off in the long run, leading to a robust and user-centric search experience.

In conclusion, testing methodologies are essential tools for ensuring software quality. By understanding and implementing black box, white box, unit, system, and acceptance testing techniques, along with a robust QA framework, you can empower your final year project to deliver an exceptional user search experience.

| **S. No.** | **Requirement ID** | **Test Case**  **Name** | **Test Case Description** | **Essential/ Desirable** | **Expected Output** | **Actual Output** | **Result** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | SRS-1 | Search Query Submission | The user should be able to enter a search query into the search bar. | Essential | Query should be submitted successfully. | Query is submitted successfully. | Success |
| 2 | SRS-2 | Search Query Processing | The search engine system should analyze the user's search query, including its keywords, intent, and context. | Essential | Query should be processed without errors | Query is processed without errors. | Success |
| 3 | SRS-3 | Relevant Search Results Retrieval | The search engine system should retrieve relevant search results from its databases, considering the user's search query and context. | Essential | A variety of relevant results should be displayed cover different interpretations of the query. | Search results are displayed. | Success |
| 4 | SRS-4, | User Context Analysis, | The search engine system should analyze the user's context, including their search history, demographics, and browsing patterns, to enhance personalization. | Essential | User’s data should be analyzed based on their previous history. | User’s data is analyzed based on their previous history successfully. | Success |
| 5 | SRS-5 | Personalized Suggestions Generation | The search engine system should generate personalized suggestions based on the user's context and the retrieved search results. | Essential | Personalized suggestions should be displayed based on the user's search history and model weights. | Personalized suggestions are displayed based on user's search history and model weights. | Success |
| **S. No.** | **Requirement ID** | **Test Case**  **Name** | **Test Case Description** | **Essential/ Desirable** | **Expected Output** | **Actual Output** | **Result** |
| 6 | SRS-6 | Personalized Search Results Display | The search engine system should combine relevant search results with personalized suggestions and present them to the user in an organized and user-friendly manner. | Essential | Personalized suggestions and search results should be displayed based on previous search history. | Personalized suggestions and search results are displayed based on previous search history. | Success |
| 7 | SRS-7 | Performance | The search engine system should provide a responsive and efficient search experience, with minimal latency and acceptable load times. | Essential | The system should remain stable and responsive. | The system remains stable and responsive. | Success |
| 8 | SRS-8 | Accuracy | The search engine system should provide accurate and relevant search results, considering the user's intent and context. | Essential | The system should retrieve a high percentage of relevant results for each query. | The system retrieves a high percentage of relevant results for each query. | Success |
| 9 | SRS-9 | Ease of Use | The search engine system should provide an intuitive and easy-to-use interface that is accessible to users of varying technical expertise. | Essential | The interface should be clear and user-friendly for managing personalized search preferences. | The interface is clear and user-friendly for managing personalized search preferences. | Success |

Table 3. Test Case

# **6. Conclusion**

In conclusion, this project has explored the unique potential of vector database tools to enhance search engine user experience through a two-pronged approach. By leveraging the inherent strengths of this technology, the project has successfully demonstrated its ability to not only personalize search queries based on individual user preferences and behavior but also act as a powerful knowledge retrieval system. This dual functionality offers a compelling path forward for search engines, promising a future where users can encounter a more intuitive, efficient, and ultimately satisfying search experience.

Furthermore, the project's findings pave the way for exciting future research endeavors. Exploring the ethical considerations of user data collection and utilization will be crucial for ensuring user trust and data privacy. Additionally, investigating the integration of advanced AI algorithms within the vector database framework holds the potential to further refine and personalize search results, creating an even more dynamic and user-centric search experience.

# **References:**

* Transformers: "Attention Is All You Need" Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin (2017). <https://arxiv.org/abs/1706.03762>

This seminal paper introduced the Transformer architecture, which has revolutionized natural language processing and many machine learning areas where sequential data plays a role.

* PyTorch: <https://pytorch.org/>

Official website of the PyTorch framework, providing extensive documentation, tutorials, and resources.

* Langchain: <https://langchain.readthedocs.io/en/latest/>
* Official Langchain documentation, including tutorials and examples.

# **Appendix**

### **Input/Output Screens:**

The below web interface highlights 2 main features:

1. Internet Search Modifier Tab:

This page consists of a single input field which is the search query itself. After clicking on the search button the output will be displayed under the output field.

1. Knowledge Retriever Tab:

This page consists of a single input field: input query. After clicking search, relevant sources will be displayed in the output fields.

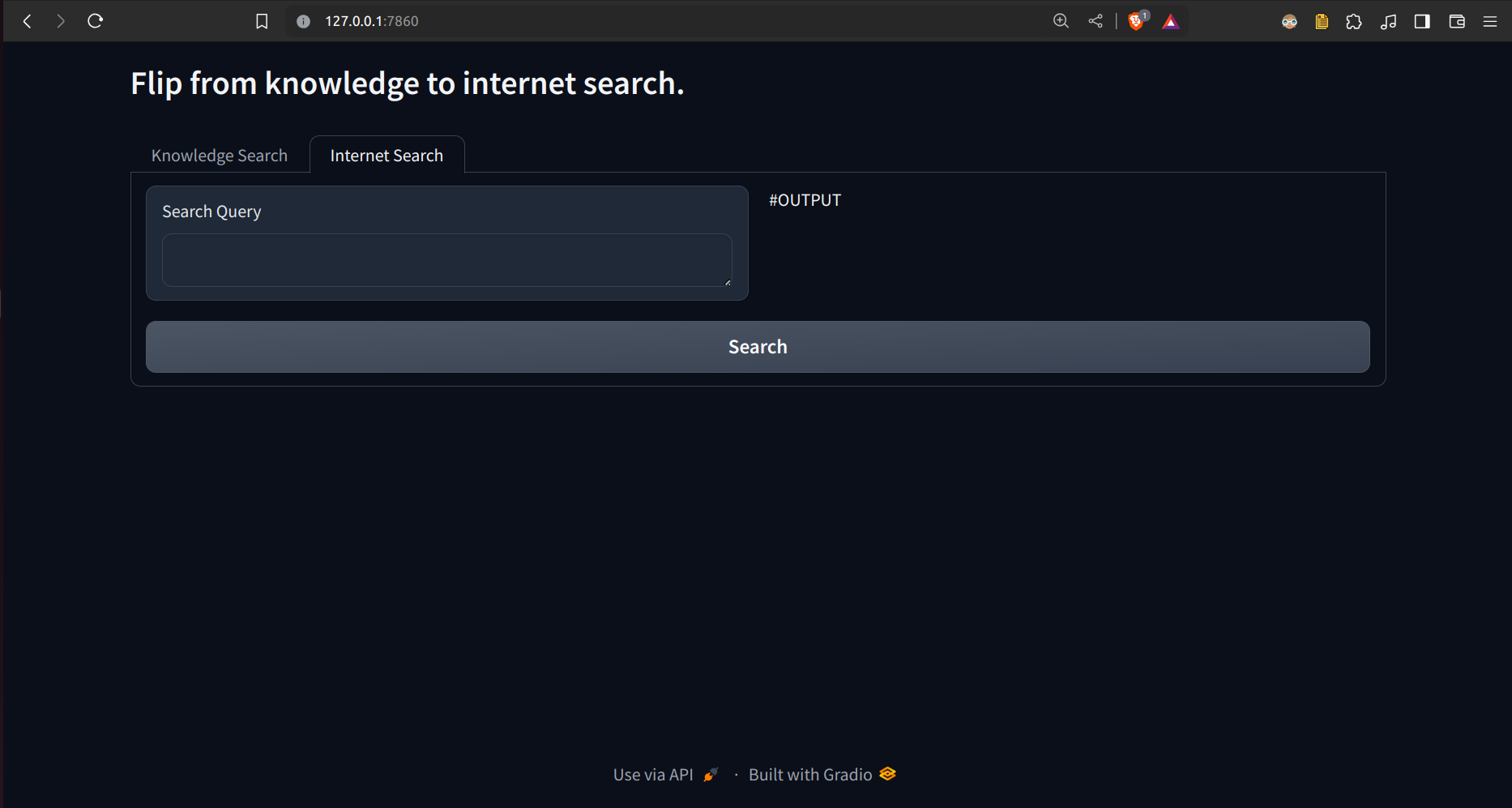


Figure 7.1 Internet Search Modifier Interface

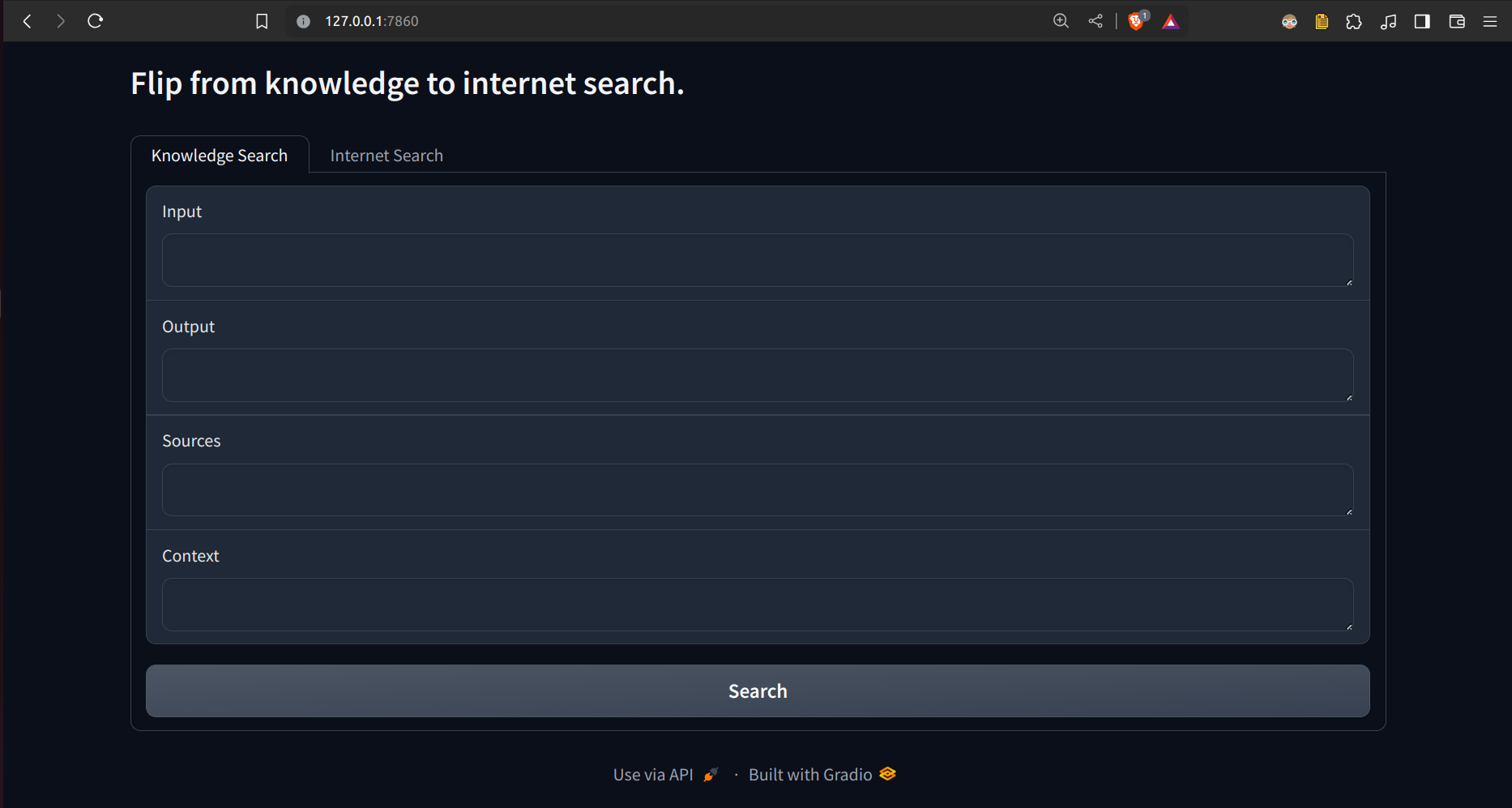


Figure 7.2 Knowledge Retriever Interface

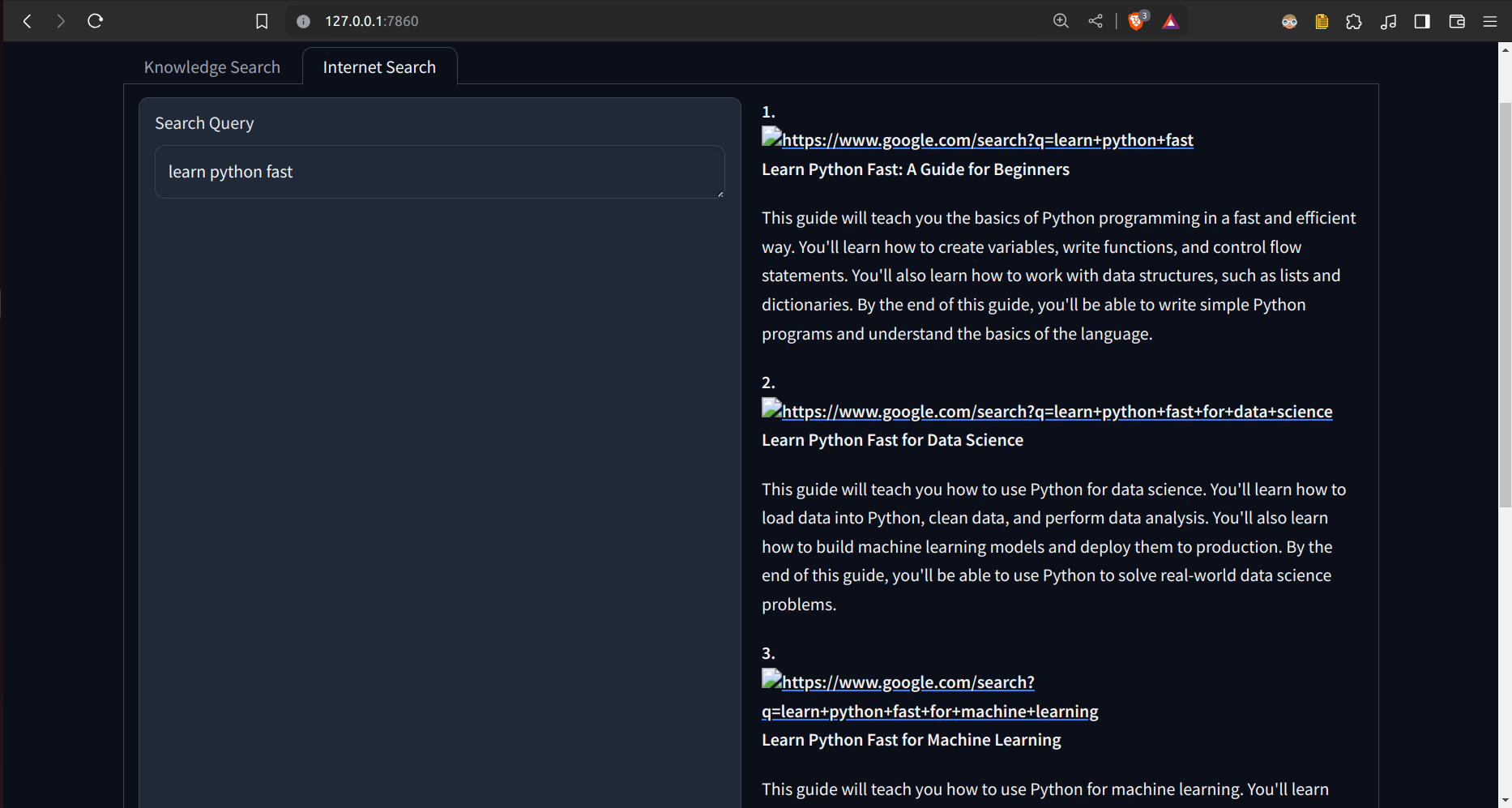


Figure 7.3 Search result query along with the output. The output field consists of 4 modified search queries along with the link and description of the modified search query.

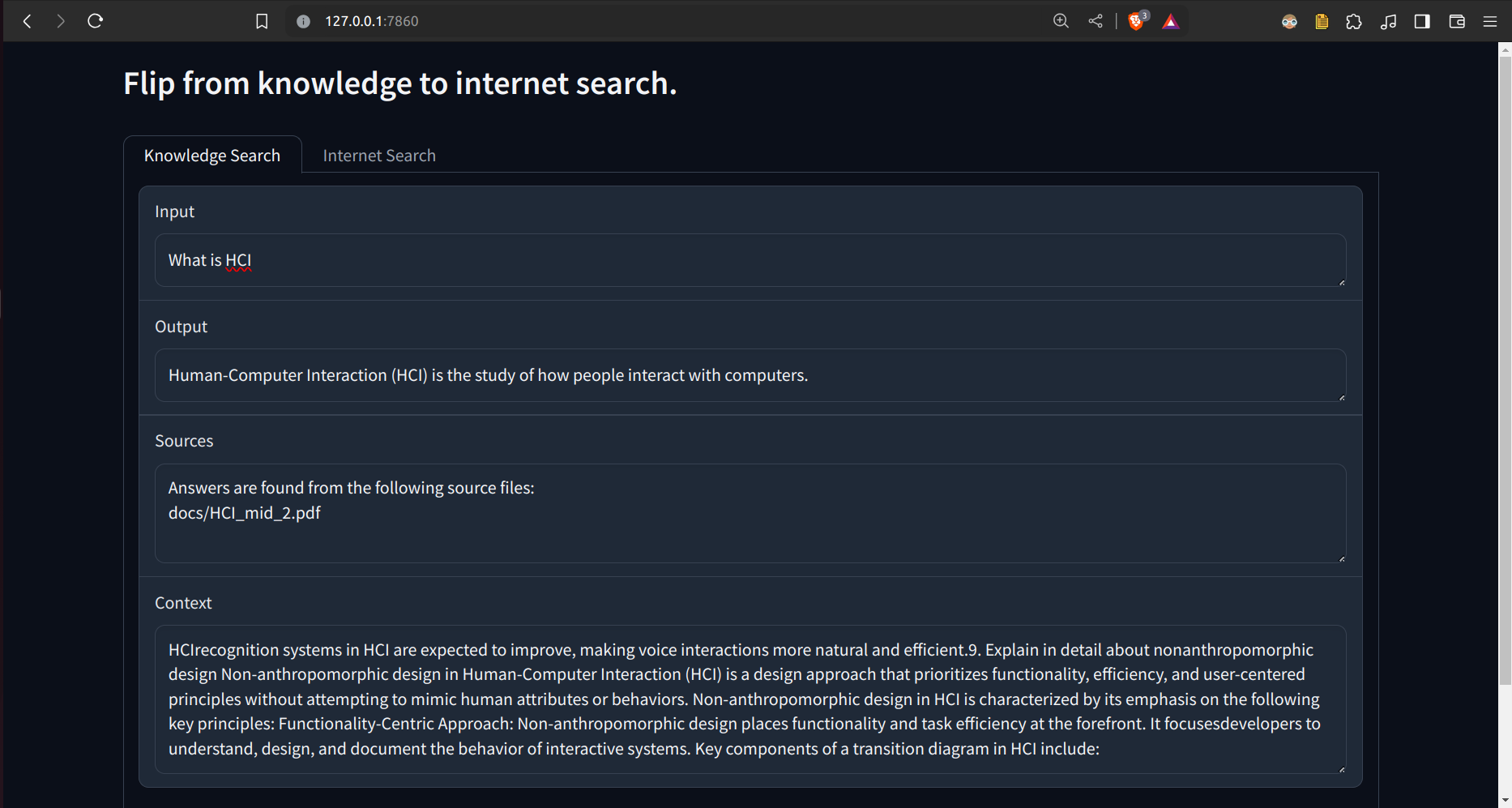


Figure 7.4 Knowledge Retriever input and output. Output fields consist of user search query answers, sources(files) from which the answer is brought, and context.

### **Sample Code:**

import requests

import os

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.embeddings import OpenAIEmbeddings, HuggingFaceEmbeddings

from langchain.vectorstores import FAISS, Chroma

from langchain.document\_loaders import DirectoryLoader

import chromadb

import gradio as gr

from llama import query\_llama2\_EP, query\_google\_API

from searchoptimize import searchModify

def initialize\_embeddings():

model\_identifier = "sentence-transformers/all-mpnet-base-v2"

print(">>>Embeddings setup completed successfully<<<")

return HuggingFaceEmbeddings(model\_name=model\_identifier)

def process\_and\_embed\_docs(dir\_path, hf\_model):

chroma\_instance = chromadb.Client()

doc\_loader = DirectoryLoader(dir\_path)

loaded\_docs = doc\_loader.load()

splitter = RecursiveCharacterTextSplitter(chunk\_size=500, chunk\_overlap=0)

split\_docs = splitter.split\_documents(loaded\_docs)

database = Chroma.from\_documents(documents=split\_docs, embedding=hf\_model)

print(">>>Embedding and chunking process completed successfully<<<")

return database

def concatenate\_documents(document\_list):

combined\_content = "".join([doc.page\_content for doc in document\_list])

print(">>>Few-shot prompting process completed successfully<<<")

print(">>>Prompt engineering process completed successfully<<<")

return combined\_content

hf = initialize\_embeddings()

# Replace the path below with the path to your dataset

example\_path = "docs"

db = process\_and\_embed\_docs(example\_path, hf)

db2=process\_and\_embed\_docs("internet", hf)

def process\_query(query):

retrieved\_docs = db.similarity\_search(query)

# print(retrieved\_docs)

sources= set()

for docs in retrieved\_docs:

sources.add(docs.metadata['source'])

print(sources)

sourceout="Answers are found from the following source files:\n"

for src in sources:

sourceout+=src+"\n"

combined\_context = concatenate\_documents(retrieved\_docs)

# print(combined\_context)

answer = query\_google\_API(combined\_context, query)

return answer.replace("\\n", "\n"), sourceout,combined\_context

def process\_internet(query):

# retrieved\_docs = db2.similarity\_search(query)

# # print(retrieved\_docs)

# sources= set()

# for docs in retrieved\_docs:

# sources.add(docs.metadata['source'])

# print(sources)

# sourceout="Answers are found from the following source files:\n"

# for src in sources:

# sourceout+=src+"\n"

# combined\_context = concatenate\_documents(retrieved\_docs)

# # print(combined\_context)

combined\_context=""

answer = searchModify(query,combined\_context)

return answer

with gr.Blocks() as demo:

gr.Markdown('''

# Flip from knowledge to internet search.

''')

with gr.Tab("Knowledge Search"):

knowledge\_input = gr.Textbox(label="Input")

knowledge\_output0 = gr.Textbox(label="Output")

knowledge\_output1 = gr.Textbox(label="Sources")

context= gr.Textbox(label="Context")

knowledge\_button = gr.Button("Search")

knowledge\_button.click(process\_query, inputs=knowledge\_input, outputs=[knowledge\_output0,knowledge\_output1,context])

with gr.Tab("Internet Search"):

with gr.Row():

internet\_input = gr.Textbox(label="Search Query")

# internet\_output = gr.Textbox(label="Output")

internet\_output= gr.Markdown("""

#OUTPUT

""")

internet\_button = gr.Button("Search")

internet\_button.click(process\_internet, inputs=internet\_input, outputs=internet\_output)

# demo.launch(share=True)

demo.launch()