VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

REPORT

CAPSTONE PROJECT

BUILDING A DOCUMENT MANAGEMENT SYSTEM

Major: Computer Science

THESIS COMMITTEE: **COMPUTER SCIENCE – CC 04**

SUPERVISOR(s): **TRAN MINH QUANG, PhD.**

BUI TIEN DUC, MSc.

REVIEWER: **PHAN TRONG NHAN, PhD.**

---o0o---

STUDENTS: **TRAN LE TRUNG CHANH (2052403)**

NGUYEN NGOC HOA (2052485)

NGUYEN NAM KHA (2052515)

HO CHI MINH CITY, MAY 2024

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

KHOA:KH & KT Máy tính
BỘ MÔN:KHMT

NHIỆM VỤ LUẬN VĂN/ ĐỒ ÁN TỐT NGHIỆP

Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: TRẦN LÊ TRUNG CHÁNH — MSSV: 2052403
HỌ VÀ TÊN: NGUYỄN NGỌC HÒA — MSSV: 2052485
HỌ VÀ TÊN: NGUYỄN NAM KHA — MSSV: 2052515
NGÀNH: KHOA HỌC MÁY TÍNH — LỚP: CC20KHM1

1. Đầu đề luận văn/ đồ án tốt nghiệp: Xây dựng hệ thống số hóa và quản lý văn bản (Building a document management system)

2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):

- Analyze the definitive features of the system and choose an appropriate architecture
- Investigate solutions to digitize paper documents and manage them such as scanning, OCR,....
- Propose a solution for data management such as ontology to manage the relationship between different documents
- Develop a system that supports document management and searching, using elastic-search
- Evaluate the proposed system

3. Ngày giao nhiệm vụ: 26/02/2024

4. Ngày hoàn thành nhiệm vụ: 19/05/2024

5. Họ tên giảng viên hướng dẫn: **Phần hướng dẫn:**

Associate Professor Tran Minh Quang, M.Eng. Bùi Tiến Đức

Nội dung và yêu cầu LVTN/ ĐATN đã được thông qua Bộ môn.

Ngày 26 tháng 02 năm 2024

CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)

PGS. TS. Trần Minh Quang

GIẢNG VIÊN HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)

PGS. TS. Trần Minh Quang

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ): _____
Đơn vị: _____
Ngày bảo vệ: _____
Điểm tổng kết: _____
Nơi lưu trữ luận án: _____

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 03 tháng 06 năm 2024

PHIẾU CHẤM BẢO VỆ LVTN

(Dành cho người hướng dẫn/phản biện)

1. Họ và tên SV:
   Trần Lê Trung Chánh — MSSV: 2052403
   Nguyễn Ngọc Hòa — MSSV: 2052485
   Nguyễn Nam Kha — MSSV: 2052515

   Ngành (chuyên ngành): Khoa học Máy tính
2. Đề tài: Building a document management system
3. Họ tên người hướng dẫn: Assoc.Prof. Trần Minh Quang and M.Eng. Bùi Tiến Đức
4. Tổng quát về bản thuyết minh:
   Số trang: Số chương:
   Số bảng số liệu: Số hình vẽ:
   Số tài liệu tham khảo: Phần mềm tính toán:
   Hiện vật (sản phẩm):
5. Tổng quát về các bản vẽ:
   - Số bản vẽ: Bản A1: Bản A2: Khổ khác:
   - Số bản vẽ vẽ tay Số bản vẽ trên máy tính:
6. Những ưu điểm chính của LVTN (Advantages):
   - Analyze and design document management system and choose an appropriate architecture
   - Investigate solutions to digitize paper documents and manage them such as scanning, OCR,....
   - Propose a solution for data management such as ontology to manage the relationship between different documents
   - Develop a system that supports document management and searching, using elastic-search
   - Evaluate the proposed system with real data
7. Những thiếu sót chính của LVTN (Weakness):
   - The ontology should be validated and improved by domain experts
   - Evaluation results should be improved
8. Đề nghị: Được bảo vệ ☑ Bổ sung thêm để bảo vệ ☐ Không được bảo vệ ☐
9. 3 câu hỏi SV phải trả lời trước Hội đồng:
10. Đánh giá chung (bằng chữ: giỏi, khá, TB): Giỏi; Điểm: 9.5/10

Ký tên (ghi rõ họ tên)

PGS.TS. Trần Minh Quang

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 27 tháng 05 năm 2024

PHIẾU ĐÁNH GIÁ LUẬN VĂN/ ĐỒ ÁN TỐT NGHIỆP

(Dành cho người phản biện)

1. Họ và tên SV: Trần Lê Trung Chánh
MSSV: 2052403 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Nguyễn Ngọc Hòa
MSSV: 2052485 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Nguyễn Nam Kha
MSSV: 2052515 Ngành (chuyên ngành): Khoa học Máy tính
2. Đề tài: Building a document management system
3. Họ tên người hướng dẫn/phản biện: TS. Phan Trọng Nhân
4. Tổng quát về bản thuyết minh:
Số trang: Số chương:
Số bảng số liệu Số hình vẽ:
Số tài liệu tham khảo: Phần mềm tính toán:
Hiện vật (sản phẩm)
5. Những ưu điểm chính của LV/ ĐATN:
- Students researched the problems in a document management system and built that system specially tailored to document digitalization using OCR technology, version control with permission, and query feature with full text search, expanded search with domain ontology, and semantic search using built-in ontology.
- The search is conducted into 3 main steps: (1) tokenization; (2) sense disambiguation; (3) refined query building. The sample application for query is law documents in Vietnamese.
- Modern technologies such as React.js, Django, PostgreSQL, ElasticSearch, Neo4j, AWS S3, Redis were employed to develop the system. The web application were deployed on AWS cloud and tested by function and front-end.

6. Những thiếu sót chính của LV/ĐATN:
- The results do not really related to the user query. The ontology building performance and large database are not managed well. In addition, an expert cannot inherit the existing ontology to develop new ones in the system.
- Some features like version control and metadata edit are not managed for consistency. Moreover, there are lack of experiments to convince why the method with LLM like Chat-GPT is the better approach.
7. Đề nghị: Được bảo vệ ☑ Bổ sung thêm để bảo vệ ☐ Không được bảo vệ ☐
8. Các câu hỏi SV phải trả lời trước Hội đồng:
a.
b.
c.

9. Đánh giá chung (bằng chữ: Xuất sắc, Giỏi, Khá, TB): Giỏi Điểm: 9.2/10

Ký tên (ghi rõ họ tên)

Phan Trọng Nhân

Declaration of Authenticity

Our team, which consists of Trần Lê Trung Chánh, Nguyễn Ngọc Hòa and Nguyễn Nam Kha, hereby declare that our Capstone Project is an authentic and original work produced by us under the guidance of Assoc. Prof. Trần Minh Quang and Mr. Bùi Tiến Đức at the Faculty of Computer Science and Engineering, Vietnam National University - Ho Chi Minh City University of Technology. We affirm that all content, ideas, methodologies, and findings presented in this capstone project are the result of our independent research, analysis, and efforts.

We further declare that:

1. Any external sources used in this project, including but not limited to books, articles, websites, and other reference materials, are appropriately cited and referenced in accordance with the academic standards and guidelines.

2. No part of this capstone project has been submitted for any other academic qualification, and it has not been previously published in any form.

3. All individuals and organizations whose work and contributions have been cited or acknowledged in this project are duly credited.

4. We have not engaged in any form of plagiarism, including the reproduction or use of the work of others without proper citation and attribution.

5. Any data, figures, charts, or images used in this project are accurate, and the sources of such data are clearly mentioned.

6. We have adhered to the ethical guidelines and standards applicable to academic research and have obtained any necessary approvals or permissions for the research conducted.

We understand that any violation of academic integrity or misrepresentation of the authenticity of this project may result in serious consequences, including the rejection of the project, academic penalties, or other disciplinary actions.

Acknowledgement

We want to express our gratitude to Assoc. Prof. Trần Minh Quang and Mr. Bùi Tiến Đức for taking the time to guide and encourage us during the planning and execution of this Capstone Project. We also appreciate the lecturers at the Faculty of Computer Science and Engineering, Vietnam National University - Ho Chi Minh City University of Technology, for providing us with the essential knowledge that enabled us to create this thesis.

Lastly, we would like to thank our families for their unwavering support throughout our university journey. Our parents' hard work serves as a major source of motivation for us. Additionally, we appreciate the guidance and training from our seniors at the company where we work. Their insights in a professional work environment have positively shaped our thoughts and behaviors.

Abstract

In the context of document management systems (DMS) in Vietnam, the prevalent reliance on physical paper documentation is proving increasingly cumbersome and inefficient. The current methodology of managing paper-based records is becoming notably challenging, adversely impacting organizational processes. The escalating volumes of paper documentation have exacerbated difficulties in maintaining organization and impede timely access to crucial information. Recognizing these limitations, there is a pressing need to transition from traditional paper-centric approaches to a digital document management paradigm.

This project aims to address the current challenges faced by DMS in Vietnam by providing a comprehensive solution for the transformation of paper documents into a digital format. The primary focus is on improving document storage and retrieval processes, ensuring a seamless transition to a more advanced and efficient system. The project emphasizes the implementation of various search approaches, including full-text search, advanced search functionalities, query expansion, and semantic search. By incorporating these elements, the goal is to empower DMS users with enhanced tools for quicker, more accurate, and contextually relevant document retrieval, ultimately providing a more streamlined and effective digital document management environment.

This thesis is thoughtfully structured over six key chapters. In chapter 1, we establish the foundation by delving into the problem, goals, scope, and related works. In chapter 2, we form the core methodology, navigating through theoretical foundations and technological intricacies. In chapter 3, we conduct a detailed analysis and design of the system, exploring stakeholders, requirements, and design components. In chapter 4, we evaluate the quality of our application through comprehensive system testing and evaluation. In chapter 5, we document the process of deployment for our system, ensuring a smooth transition from development to operational status. Conclude in chapter 6, we discuss potential improvements for our system and provide a conclusive summary of the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In Chapter 1, we lay the groundwork by presenting a concise problem statement, articulating our goals, defining the project's scope, and reviewing related works as well as pointing out the advantages and disadvantages of their systems. This initial chapter sets the stage for our unique approach to addressing the identified challenges.

1.1 Problem statement

In the transformative era of Industry 4.0, where the seamless utilization of digital resources is pivotal for the success of businesses and organizations, the importance to efficiently manage documents has never been more pronounced. In the dynamic landscape of digital document management systems, the folder hierarchy for organizing files, while providing a semblance of order, poses significant user challenges and inefficiencies. Hence, there's a pressing need for a paradigm shift in how we handle information organization and retrieval in the dynamic realm of digital document management systems.

One of the major problems users face is forgetting where they stored important documents. This not only leads to frustrating searches but also affects productivity. Users often get lost in the maze of folders, struggling to remember where a specific document is saved. Remembering these complex file paths adds an extra layer of difficulty to daily tasks, taking up time and mental energy. As the digital world expands, finding innovative solutions to this challenge becomes increasingly important for improving user experience and workflow efficiency.

Another part of this problem is that people struggle to decide where to put new documents in the first place. Users often feel unsure and overwhelmed when choosing the right folder for their files. This not only slows down the process of organizing documents but also raises questions about the logic and user-friendliness of the folder system. Figuring out the best way to organize files becomes a big hurdle in making document systems efficient. Users, faced with many folders and subfolders, might hesitate or doubt their decisions, making the filing process even more complicated. This lack of clarity in document organization not only delays finding

information but also highlights the need for better and user-friendly design to make navigating through digital filing systems less mentally taxing. Solving this complex challenge requires a comprehensive approach that makes finding documents easy and helps users make smart choices when organizing them.

Another intricate facet of the overarching challenge lies in the less-than-optimal performance of the search functionality. Despite technological improvements, users often get search results that don't match what they're looking for. This inconsistency adds an element of unpredictability and makes the system less reliable, causing frustration and longer search times. Users expect a quick and accurate search, but when the results don't deliver, they might resort to manually searching through folders, making the issues of forgetfulness and uncertainty even worse. Addressing this aspect of the issue necessitates a nuanced approach that not only enhances the precision of search algorithms but also aligns them more closely with user intent, thereby fostering a more reliable and efficient document retrieval experience.

Putting all these issues together shows a clear need for rethinking and improving document management systems. We need a solution that overcomes the problems of the traditional folder system while making sure users can easily find what they need. By exploring and suggesting new ideas, we aim to change how document management works, creating a system that fits seamlessly with what users want and expect.

1.2 Goals

This thesis presents the development of an advanced document management system designed to streamline the entire document handling process, from upload to retrieval. The primary objective is to create a seamless experience for users, enhancing their ability to manage documents efficiently. Key features of the system include:

- **Seamless Document Upload and Metadata Extraction**: Users can upload documents without the need to categorize them into specific folders. The system automatically performs Optical Character Recognition (OCR) and extracts metadata from the uploaded documents, simplifying the upload process and ensuring all necessary information is captured.
- **Flat Structure for Ease of Access**: The system employs a flat structure, eliminating the need for users to decide on storage locations for their documents. This approach minimizes the effort required to organize documents and simplifies retrieval.
- **Comprehensive Document Management**: The system includes robust features for man-

aging digital documents, such as version control and access permissions. Users can track document versions to manage changes effectively, and control who has access to specific documents, ensuring security and maintaining document integrity.

- **Effective Document Retrieval Methods**: To facilitate quick and efficient document retrieval, the system offers various advanced search methods. Full-text search capabilities allow users to locate documents based on their content, while semantic search enhances this by understanding the context and meaning behind search queries.
- **Integration of Ontology for Improved Search Accuracy**: Integration of Ontology for Improved Search Accuracy: The quality of full-text searches is further improved through the integration of an ontology. This ontology acts as a knowledge base, refining user queries and providing more accurate search results.

Through these features, the document management system aims to significantly improve user experience in document management, making it simpler, more intuitive, and highly efficient.

1.3 Scope

Considering the diverse range of document management systems available in the market, each serves a different purpose, the scope of this project has been determined to include the following specific characteristics:

- **File format**: The system is specifically designed to store documents that are scanned from physical paper. Given that these scanned documents are typically in the `pdf` format, our system has been optimized exclusively to support the storage and management of `pdf` files only.
- **Documents domain**: The system is designed to serve diverse domains of documents, tailored to the specific needs of agencies and businesses. Our system's functionalities, such as ontology building and metadata extraction, are customizable to meet the requirements of these entities. In this thesis, we have selected the domain of Vietnamese legal normative documents as our primary focus for two key reasons. Firstly, there is a substantial volume of Vietnamese legal documents accessible from various sources, facilitating ease of collection. Secondly, the legal domain encompasses a wide range of topics from multiple other domains, allowing us to utilize this characteristic to evaluate our system's functions across diverse areas.
- **Building ontology**: The ontology functions as a comprehensive knowledge base tailored

to a specific domain, necessitating expert knowledge within that field. This thesis focuses on methodologies to utilize ontology structures to enhance search results, yet the creation of expert-driven ontologies remains a significant challenge. Consequently, our project includes an interface designed for expert users to construct these ontologies. The default ontology developed within this project is an upper ontology, encompassing broad and general knowledge, which may not be suitable for specialized documents. Effective retrieval for such documents necessitates the use of specialized ontologies crafted by domain experts to optimize search outcomes.

Chapter 2

Related works

Chapter 2 explores existing document management systems, focusing on their functionalities, advantages, and drawbacks. This chapter provides a critical review of related works to identify current trends and challenges in the DMS landscape. By examining various systems, we gain insights into their design philosophies, integration capabilities, and user experiences. This analysis serves as a foundation for developing an enhanced DMS that addresses the limitations identified in existing solutions, particularly in the areas of search functionality and document retrieval. The goal is to leverage these insights to create a system that not only meets but exceeds user expectations in managing and accessing documents efficiently.

2.1 VNPT-eOffice

VNPT eOffice [1] is an outsourced Vietnamese DMS developed by Vietnam Posts and Telecommunications Group (VNPT). It provides comprehensive document management functionalities, including document capture, storage, workflow automation, and digital signature support.

VNPT eOffice, the most popular DMS in Vietnam, is specifically designed to meet the requirements of Vietnamese businesses and government agencies, adhering to local regulations and data privacy standards.

Key Features of VNPT- eOffice:

- Automatic document capture and indexing
- Document capture and storage
- Workflow automation and approval processes
- Electronic signatures and document certification
- Find documents quickly and easily using keywords, tags, and advanced search filters

- Integration with Vietnamese government platforms

Pros:

- Tailored for specific organizations and businesses.
- Well-integrated with Vietnamese government platforms.
- Backed by VNPT's infrastructure, ensuring data security and system uptime.

Cons:

- Might not be suitable for organizations with international operations.
- Basic search functionalities: While effective for simple searches, might lack advanced features for complex document retrieval tasks.
- Vendor lock-in: Switching to another DMS later might be challenging due to data migration complications.

From our observation, VNPT eOffice stands out as a tailored and evolving outsourced Document Management System (DMS) for Vietnamese businesses, offering customizable features aligned with individual needs. It is mainly designed for handling documents inside the scope of businesses, VNPT eOffice provides search filters specific to Vietnamese government documents and regulations, catering to the needs of local businesses and organizations. Although, currently, VNPT eOffice only offers basic search functionalities, allowing users to search documents based on keywords, document type, author, and date range.

2.2 M-Files

M-Files [2] is a popular enterprise DMS known for its intuitive interface and powerful metadata management capabilities. It utilizes intelligent indexing to automatically extract metadata from various document formats, making it easy to find relevant information.

Key Features of M-Files:

- Automatic metadata extraction
- Powerful search functionalities with advanced filters
- Version control and audit trails
- Secure document storage and access controls

- Workflow automation tools

Pros:

- Automatic metadata extraction and classification eliminates manual tagging
- Metadata-driven organization provides a more intuitive way to find and access information
- M-Files reduces manual tasks and streamlines workflows by using AI, leading to increased employee efficiency.

Cons:

- M-Files is more expensive compared to some basic DMS solutions.
- Though flexible, M-Files might not offer the level of granular customization desired by some users.
- Not ideal for highly structured environments: M-Files thrives on flexible content, and highly structured data might require adjustments for optimal performance.

M-Files stands out for its strong focus on effective document management. It makes finding documents a breeze by using smart techniques to organize information based on metadata. Users can search using keywords, metadata fields, and document properties. What's even more impressive is the variety of filters available, like date range, document type, author, and custom metadata fields. This not only makes searching easy but also shows M-Files' dedication to making things work seamlessly for its users. In a nutshell, M-Files is a solid and user-friendly document management solution, designed to adapt to the unique needs of its users, its strengths lie in its intelligent search, unified content management, and workflow automation capabilities.

2.3 Folderit

Folderit DMS is a cloud-based document management system (DMS) known for its user-friendly interface and focus on document security. It offers features like file sharing, version control, and access control to manage and collaborate on documents effectively.

Key Features of Folderit:

- Metadata tagging
- Custom Metadata File Linking

- Workflow Automated Retention
- Version control and audit trails
- File sharing and collaboration
- Strong document security: Bank-level security with triple backups and encryption.

Pros:

- Simple and affordable: Cost-effective option for small businesses and individuals
- Secure and reliable: Strong security features and cloud storage provide peace of mind
- Customizable metadata fields, allowing for easy linking and template definition at the folder level
- Search on content of the files thanks to Optical Character Recognition (OCR).

Cons:

- May lack advanced functionalities compared to enterprise-grade DMS solutions.
- Customization options might be limited compared to some competitors.
- Limited workflow automation and AI features compared to some DMS solutions.

Overall, Folderit DMS is a suitable option for small businesses and individuals seeking a simple and secure document management system with a low price point. However, users needing advanced features, extensive metadata capabilities, or significant scalability might benefit from exploring other DMS solutions. Moreover, basic search capabilities may hinder thorough data exploration, posing challenges for organizations dependent on complex document retrieval.

2.4 Conclusion

In summary, our investigation culminates in a table encapsulating the strengths and focal points of each platform. We abstain from labeling any aspect as ”weak” or ”bad” as companies do not openly display weaknesses on their websites. Additionally, delivering a thorough review for each platform necessitates a significant time investment beyond our current capacity. Consequently, Table 2.1 functions as a reference sheet, spotlighting aspects from these solutions that we can gain insights from.

Table 2.1: Comparison of related DMS

	VNPT-eOffice	**M-Files**	**Folderit**	**Our System**
Document organization	Folder	Flat	Folder	Flat
Metadata extraction	Limited	Automatic		Automatic
Advanced search	Yes	Yes	Yes	Yes
Search query expansion	No	No	No	Yes
Semantic search	No	No	No	Yes
Version control	Yes	Yes	Yes	Yes

Looking at the summary Table 2.1, it's clear that different DMS solutions cover the basics for businesses to organize and find documents. Additionally, most companies also have special features to stand out. Especially, M-Files takes things to another level with its AI-powered features to extract metadata automatically and analyze documents.

However, within the project's scope, we don't plan to compete with various top-notch features of M-Files considering the time and budget constraints, instead we want to enhancing and addressing its existing limitations: searching. Our system's main focus is making it easy for users to find documents by employing various search methods that cover a wide range of scenarios. Even though we can't fully automate extracting details as M-Files, we still do some by providing support for legal documents, one step ahead of VNPT-eOffice, one of the most popular DMS being used in Vietnam.

Chapter 3

Methodology

In Chapter 2, we offer a comprehensive exploration of the theoretical basis and technologies underpinning our solution. We provide a comprehensive overview of the chosen technologies, explaining the rationale behind each selection and highlighting their respective advantages and disadvantages. This includes discussions on web user interface development, server development, database management systems, security measures, OCR technology, cloud storage solutions, and search engines, with a specific emphasis on semantic search. The chapter strategically aligns theoretical insights with practical technological choices, laying a solid foundation for the project's development and implementation.

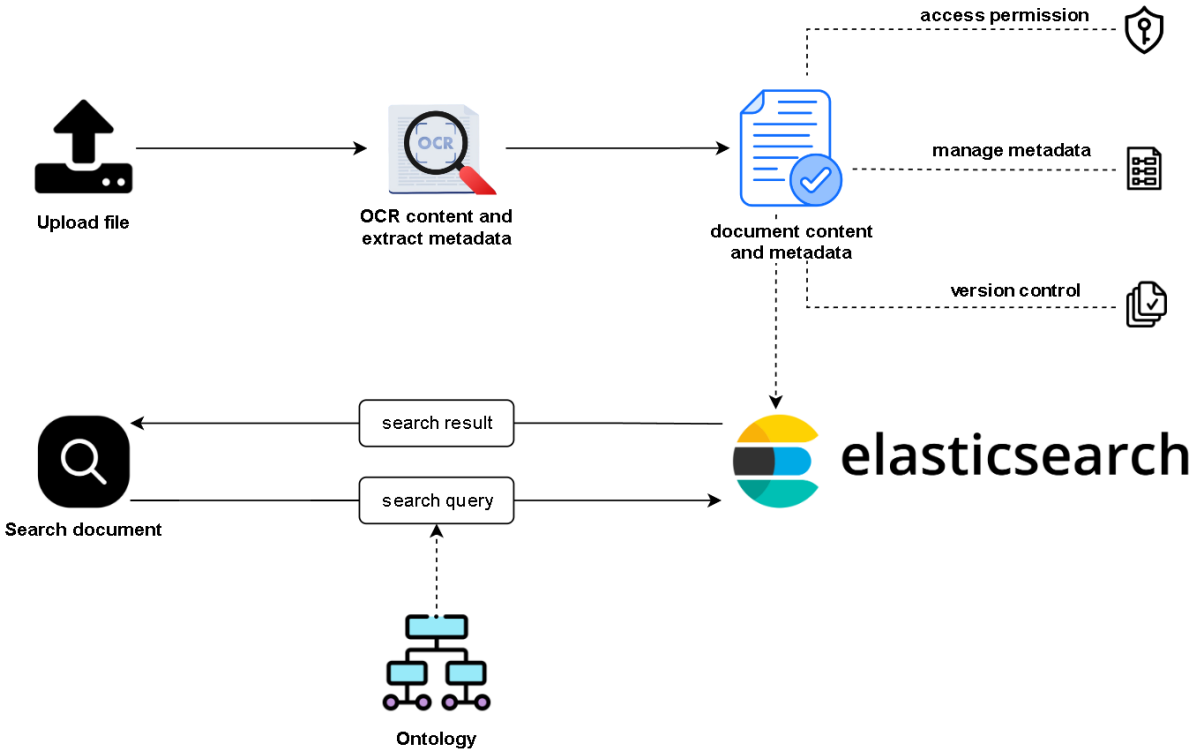

Figure 3.1: Overall system

Figure **3.1** illustrates the overall architecture of our system, which comprises several inter-

connected components designed to streamline the document handling process from upload to retrieval. The main components include:

- **File Upload Interface**: A user interface that facilitates the uploading of documents into the system.
- **OCR and Metadata Extraction**: A module that processes uploaded documents, extracting textual content and metadata using OCR technology.
- **Document Management**: Allows users to perform CRUD (Create, Read, Update, Delete) operations on documents, manage access permissions, metadata, and version control.
- **Elasticsearch Integration**: A backend component where extracted content and metadata are indexed for efficient search and retrieval.
- **Search Interface** : A user interface that enables users to query the documents and retrieve relevant results using various search methods offered by the system.
- **Ontology Integration**: Enhances search functionality by using ontologies to improve the accuracy and relevance of search results.

3.1 Theoretical basis

3.1.1 Ontology

In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. It helps to clearly outline how these elements are related within a particular field or across multiple areas of study. The components of ontology are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). [3]. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of terms and relational expressions that represent the entities in that subject area. Ontology is essential in knowledge representation because it provides a clear and organized framework. It acts as a foundation for detailing the complexities of a specific domain. By defining concepts, properties, and their relationships, ontology captures the essence of knowledge, creating a well-structured and coherent system.

One of the pivotal functions of ontology is to define the fundamental concepts within a given knowledge domain. These concepts represent the building blocks of understanding and are connected through well-defined relationships. For example, in a medical ontology, concepts

could range from diseases to treatments, and relationships could denote associations such as ”treated by” or ”causes.” Another distinctive strength of ontology lies in its ability to foster a shared language for communication. By establishing a common vocabulary and formalizing the semantics of concepts, ontology becomes a bridge that facilitates effective connections among diverse objects, especially documents. This shared language transcends human comprehension barriers and extends to machine interpretation, creating a harmonious intersection of human and systems.

Types of ontology

Different types of ontology share many structural similarities, regardless of the language in which they are expressed. Most ontologies describe individuals (instances), classes (concepts), attributes and relations. There are various types of ontology, each serving different purposes and levels of detail, two of them are ***domain ontology*** and ***upper ontology***.

a. Domain ontology

A domain ontology, or domain-specific ontology, encompasses concepts specific to a particular field, like biology or politics. These ontologies usually define terms within that specific context. They standardize vocabulary, support data interoperability, and enhance information sharing and retrieval by establishing a shared understanding of the domain. Some famous domain ontologies include:

- **SNOMED CT** [4]: An extensive, multilingual clinical healthcare terminology that provides codes, terms, synonyms, and definitions used in clinical documentation and reporting.
- **GoodRelations** [5]: This ontology is used to describe products, prices, company profiles, and other e-commerce-related information. It enables data integration and better search engine optimization (SEO) for e-commerce websites by providing a structured and standardized way to present product information.
- **MASON (Manufacturing Service Ontology)** [6]: Defines the concepts related to manufacturing processes, resources, and capabilities, enabling better integration and communication across different manufacturing systems.

Since different people create domain ontologies, they describe concepts in unique and specific ways, often making them incompatible for the same project. As systems using these ontologies expand, they frequently need to merge them by manually adjusting each part or utilizing a combination of software tools and manual efforts. This presents a challenge for the ontol-

ogy designer. Different ontologies within the same field arise due to variations in language, intended use, and perspectives on the domain, influenced by factors such as culture, education, and beliefs.

Currently, merging ontologies that are not based on a common upper ontology is largely done manually, which is time-consuming and expensive. However, when domain ontologies use the same upper ontology to define their entities, they can be integrated more easily. There are studies on generalized techniques for merging ontologies, such as Dynamic Ontology Repair [7] but this area of research is still developing.

b. Upper ontology

Upper ontology, also referred to as a top-level ontology, upper model, or foundation ontology, is a type of ontology that includes very general terms like ”object,” ”property,” and ”relation,” which are applicable across all domains. Its primary role is to enhance broad semantic interoperability among numerous domain-specific ontologies by offering a common basis for defining terms. In this structure, terms in the domain ontologies are organized under the terms in the upper ontology, making the upper ontology classes the superclasses or supersets of all the classes in the domain ontologies.

A number of upper ontologies with its own proponents have been proposed, which are:

- **BFO (Basic Formal Ontology)** [8]: A top-level ontology which was built for the purposes of promoting interoperability among domain ontologies built in its terms through a process of downward population. The structure of BFO is based on a division of entities into two disjoint categories of continuant and occurrent, the former consists of objects and spatial regions, the latter contains processes conceived as extended through (or spanning) time. BFO thereby seeks to consolidate both time and space within a single framework.

- **SUMO (Suggested Upper Merged Ontology)** [9] : an upper ontology intended as a foundation ontology for a variety of computer information processing systems. SUMO defines a hierarchy of classes and related rules and relationships. It focuses on meta-level concepts (general entities that do not belong to a specific problem domain), and thereby would lead naturally to a categorization scheme for encyclopedias

- **WordNet** [10]: developed by Princeton University, Wordnet is a freely available database originally designed as a semantic network based on psycholinguistic principles, was expanded by addition of definitions and is now also viewed as a dictionary. It qualifies as an upper ontology by including the most general concepts as well as more specialized concepts, related to each other by relations such as hypernym, hyponym, part-of and cause.

Ontology in our digital document management system

Considering our document management system which may stores documents spanning diverse domains, the creation of domain-specific ontologies is a hard challenge due to our limited expertise in these domains. Consequently, our approach is to develop a cross-domain upper ontology in order to serve the needs of general users within the system. The structure of our ontology resembles the Princeton Wordnet's structure, as depicted in **Figure 3.2**

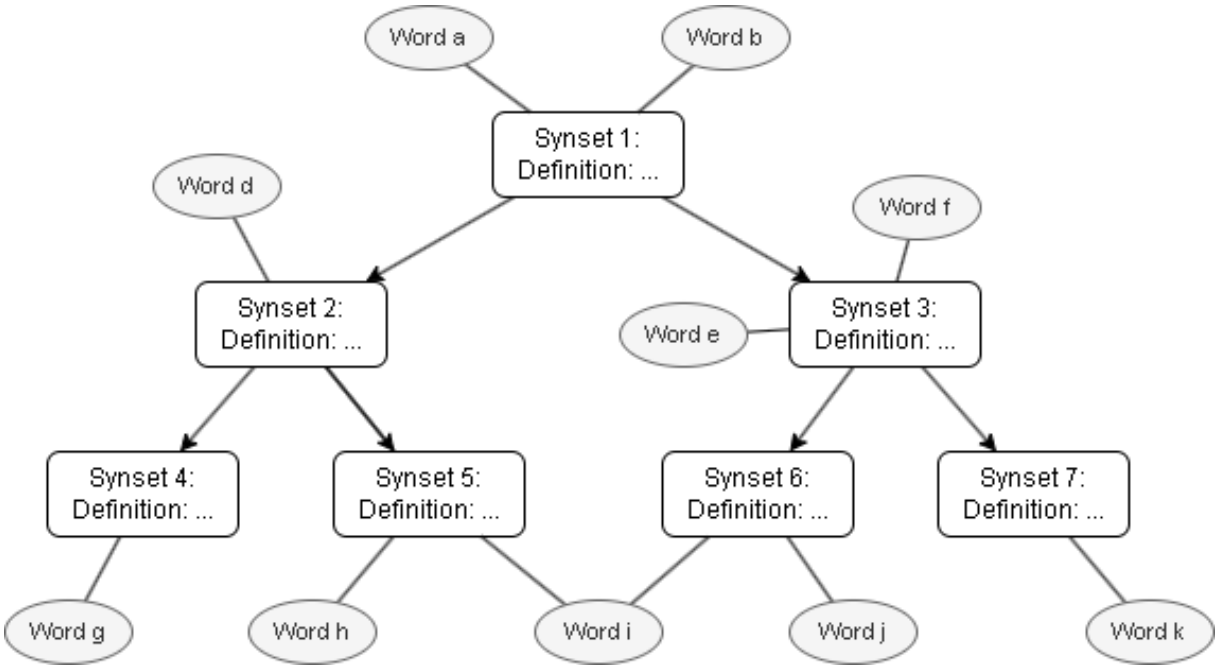

Figure 3.2: Structure of the ontology

There are two types of nodes in this ontology: `Synset` and `Word`, along with two types of relationships: `HAS_HYPONYM` (directed edges) and `BELONGS_TO` (undirected edges). `Words` are grouped into sets of cognitive synonyms (`Synset`), each expressing a distinct concept. Synonyms - words that are grouped into one synset - denote the same concept and are interchangeable in many contexts. The undirected relationship `BELONGS_TO` is used to represent this connection between `Words` and `Synset`. It shows which `Word` belongs to which `Synset`. Word with several distinct meanings are represented in as many distinct synsets, as shown in **Figure 3.2**, *Word i* belongs to both *Synset 5* and *Synset 6*. For example, the word *bank* usually means *"an organization that provides various financial services"*. However, it also has another meaning that is *"the side of a river, canal"*. Consequently, it is included in two separate synsets, each capturing a different meaning of the word.

Each `Synset` has an attribute called "Definition", which provides a brief description of the concept represented by the Synset. Additionally, usage examples are included to illustrate how

the members of the Synset are used. `Synset` are organized into hierarchies, defined by the `HAS_HYPONYM` relationship. It links more general synsets like {*vehicle*} to increasingly specific ones like {*car, automobile*} and {*bike, motorcycle*}. The directed relationship `HAS_HYPONYM` is used to show this connection between two synsets. In this relationship, the source is the synset with a broader (more general) meaning, and the target is the synset with a narrower (more specific) meaning.

This hierarchical structure of the ontology is deliberately designed to align with the foundational theories of human semantic memory established in the 1970s [11]. Experiments indicated that humans organized their knowledge of concepts in an hierarchical fashion, with more general categories at the top and increasingly specific ones below. By aligning with theories of human semantic memory, this ontology structure can reflect how people naturally organize and conceptualize information, thereby support them in searching for documents. When a user searches for a particular term, the system can leverage this ontology hierarchical structure to suggest related keywords that are either broader (higher in the hierarchy) or more specific (lower in the hierarchy) than the search term. This enhances the user experience by making it more intuitive and reflective of how humans naturally think about concepts. Users are more likely to find the suggested keywords relevant and useful because they correspond to the way concepts are mentally organized, leading to better recall and understanding. Then, users can explore related concepts and refine their search queries based on the suggestions provided by the system, ultimately leading to more accurate and efficient search results.

3.1.2 Query expansion

Query expansion [12] is a refined information retrieval technique employed within search engines and databases to enhance search result precision. This method involves the augmentation of an initial user query through the systematic inclusion of semantically related terms, synonyms, or conceptually connected terms. The final objective is to broaden the scope of the search query, thereby mitigating potential issues related to term mismatch or the inherent ambiguity of user inputs.

The theoretical basis lies in the assumption that users may not always express their information needs accurately or completely in a short query. By adding related terms or synonyms to the query, the chances of retrieving relevant documents that might have been missed initially can be increased.

In the scope of our project, we restrict the application of query expansion solely to the incorporation of relevant terms into the original query through the combination of boolean operator ”OR” between the original term and the expanded term.

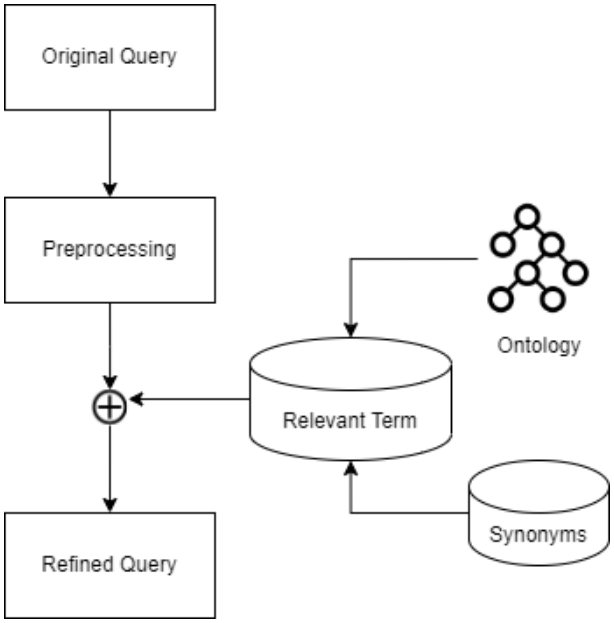

Figure 3.3: Process of creating an expanded query

The workflow is as **Figure 3.3**, the initial step involves parsing the user query to extract specific terms, which are then identified as keywords. Subsequently, we systematically generate a list of potentially related terms for these keywords sourcing from the specific domain ontology and integrate them into the query, broadening its scope. Additionally, users are empowered to further refine their search by incorporating more relevant terms, which may have a broader or narrower meaning compared to the keyword, suggested from the ontology we have developed. This iterative process ensures a dynamic and collaborative approach to query expansion, enhancing the precision and inclusivity of information retrieval, while still making sure that the user can select the preferred terms to add to the query instead of automatically adding all relevant terms to the user query, which can cause the search results filled with irrelevant information.

Query expansion holds particular relevance in scenarios where users may express information needs using varied terminology or where the underlying dataset contains a diversity of linguistic expressions for a given concept. By incorporating a broader spectrum of relevant terms, query expansion aims to bridge the gap between the user's intended information needs and the documents available in a retrieval system, ultimately enhancing the effectiveness of information retrieval.

3.1.3 Semantic search

Semantic search [13] marks a significant transformation in information retrieval, departing from conventional keyword-centric methods to understand the intent behind user queries. The

conceptual basis of semantic search is rooted in the domains of natural language processing and artificial intelligence.

Benefits of semantic search:

- More relevant results: Users get what they're actually looking for, even if they don't use the exact keywords.
- Improved user experience: Searching becomes more intuitive and efficient.
- Discovery of new information: Users can find related topics and concepts that they may not have been aware of.

In our project, we want to make the most of semantic search to find what the user is looking for in a large pool of documents. Our main goal is to use a ready-to-go semantic search method that goes beyond just the user's question but the complexity and practicality of this method still fit well within the project's limits.

There are some approaches used in semantic search to achieve its goals of understanding user intent and delivering relevant results.

a. Knowledge graph

Knowledge graphs [14] represent entities and relationships, creating a structured semantic framework. Construction involves defining ontologies and using triple stores to store subject-predicate-object relationships. Graph-based algorithms enable efficient traversal for context-aware results.

Pros:

- Provides explicit relationships between entities.
- Enhances contextual understanding through graph traversal.
- Enables the incorporation of domain-specific knowledge.

Cons:

- Construction and maintenance of knowledge graphs can be resource-intensive.
- Dependency on the quality and completeness of underlying data.

From our point of view, creating and keeping up these graphs demands a lot of time, expertise, and a solid grasp of the specific subject. Moreover, it relies heavily on having accurate and

complete data, which might be tricky in situations where information is changing or incomplete. The inflexible nature of knowledge graphs can make them less adaptable to rapidly changing data and may struggle with diverse user queries, which is a problem considering our system is expected to handle general documents from various domains. Considering these challenges, we decided to explore a more practical approach for our system.

b. Natural language processing (NLP)

NLP approaches are centered on comprehending the semantics of natural language. Through advanced techniques, words are mapped into high-dimensional vector spaces, facilitating the capture of complex and context-dependent semantic relationships. The utilization of deep learning models, which consider the entire context of a word within a sentence, enhances the system's ability to discern nuanced language nuances, including how individual words interrelate.

Pros:

- Captures complex and context-dependent semantic relationships in natural language.
- Adaptable to various languages and domains.
- Allows for the interpretation of user intent and context.

Cons:

- Requires substantial computational resources for training and inference if hosting on-prem.
- May face challenges with out-of-domain or specialized vocabulary.
- Interpretability can be challenging due to the complexity of the model architecture.

In the context of our system, using natural language processing (NLP) methods, more specifically word embeddings method to represent semantic meaning, is the appropriate choice for the system. While we acknowledge the challenges in computing power and interpreting results, the widespread availability of cloud-based solutions, mitigates the issue of resource-intensive training and inference. Additionally, leveraging pre-trained models such as those provided by OpenAI's plug-and-play embedding model simplifies the implementation process and reduces the need for extensive domain-specific training data. This approach enables us to efficiently capture semantic relationships across various languages and domains, enhancing the interpretability of user queries and facilitating more accurate information retrieval.

3.2 Technologies

3.2.1 Web user interface development

The frontend of the system acts as the link between users and the system's business logic, facilitating user interactions. Users have to spend a significant amount of time engaging in tasks like creating, manipulating, and receiving data from the server. Therefore, it's crucial for the frontend to be user-friendly, interactive, and responsive. This focus ensures a seamless and engaging experience for users interacting with our digital document management system.

Including various essential functionalities such as user management, document handling, ontology building, and advanced search capabilities as well as other technical considerations, the choice of React for our digital document management system's front-end is a thoughtful decision based on its ability to handle these requirements.

React is a widely-used JavaScript library for building user interfaces, particularly for creating interactive and dynamic web applications. Developed and maintained by Meta, React has gained popularity for its efficiency and flexibility in constructing reusable UI components. It stands out as a compelling choice due to its unparalleled component-based architecture, enabling modular development and facilitating code reuse. This modularity enhances maintainability, scalability, and collaborative development efforts, aligning perfectly with the complexity of our document management system. React's virtual DOM implementation further contributes to enhanced performance, ensuring rapid updates and rendering of UI components, which is critical for real-time document management.

Another major factor that led us to choose React over alternative frameworks such as Vue or Angular is its widespread community support and extensive ecosystem. The vast collection of pre-built components, libraries, and third-party integrations available within the React ecosystem accelerates development, reduces time-to-market, and ensures compatibility with emerging technologies.

Additionally, React's unidirectional data flow and one-way binding simplify state management, fostering predictable behavior and debugging. This architectural decision enhances the overall robustness of our system, particularly in handling complex operations related to user management and ontology building.

Furthermore, the flexibility provided by React in terms of integrating seamlessly with other libraries and frameworks aligns with our goal of creating a dynamic, feature-rich document management system. The ability to effortlessly incorporate advanced functionalities, such as real-time collaboration and efficient document versioning, positions React as a strategic choice

for the front-end development of our project.

We aim to not only meet the functional demands of document management but also enhance the overall user experience, through a modular architecture, performance optimization, and a commitment to user-friendly design, React has shown us to be an excellent choice for the front-end development of our digital document management system.

3.2.2 Server development

In the realm of online applications, there exists a crucial infrastructure comprising a server and a database that operates behind the scenes of the frontend. The server serves as the central processing unit, acting as the system's "brain." It acts as an intermediate layer between the frontend and the database, receiving requests from the frontend and performing operations on the associated data stored in the database.This pivotal component of software development necessitates careful planning and meticulous structuring. It is in this domain that software architecture practices truly excel, enabling the creation of a well-organized and efficient system.

a. Requirements for our server

The server infrastructure requirements for our project, dedicated to constructing a digital document management system featuring OCR technologies, metadata extraction, ontology-based relationships, and search functionalities, are integral to the success and efficiency of our solution. To ensure flexibility and customization, the server must be designed with modularity and a service-oriented architecture, incorporating distinct modules for OCR, metadata extraction, ontology management and search capabilities including content search, semantic search and ontology-based search. This modular approach not only fosters adaptability but also inherently bolsters system resilience, allowing independent operation of services to minimize the impact of failures or updates on the overall system.

In addition to the aforementioned requirements, it is crucial that the server for the digital document management system is designed to be user-friendly and easy to set up and deploy. This user-friendly approach is particularly important as it caters to customers in the waste collection industry who may possess more familiarity with machinery than with technology. By prioritizing simplicity and ease of integration into existing workflows, the system can be quickly adopted by customers with varying levels of technological expertise.

We also desire our server to be scalable by default, especially horizontal scalability. By implementing a horizontally scalable server infrastructure, we can seamlessly handle a growing number of documents, users, and concurrent operations. This scalability ensures that our digital document management system remains responsive and efficient, even during peak usage periods

or when experiencing sudden surges in demand.

b. Django

Django [15], a powerful web framework written in Python, is an excellent choice for building the server infrastructure to meet the requirements of the digital document management system. It aids developers in building high-performance web applications that can manage massive volumes of content.

Django's exceptional modularity and service-oriented architecture align seamlessly with our project's imperative need for flexibility and customization in developing a cutting-edge digital document management system. This framework empowers us to disassemble the system into independent components, treating OCR, metadata extraction, ontology management, and document relationship establishment as separate Django applications. This approach not only facilitates agility and adaptability but also inherently reinforces system resilience by insulating one component's failures or updates from significantly impacting the overall system.

The inherent user-friendly nature of Django positions it as an ideal solution tailored to the needs of our target audience involved in the document management process. With Django's clear and intuitive structure, the system becomes effortlessly set up and deployed. Its extensive documentation and robust community support ensure accessibility, enabling users with varying levels of technological expertise to swiftly integrate the system into their existing workflows.

Scalability, a critical concern for our digital document management system, finds an exemplary solution in Django's architecture. The framework inherently supports horizontal scalability, allowing us to effortlessly manage an increasing influx of documents, users, and concurrent operations. Leveraging Django's capabilities in distributed computing and load balancing, our system gains the ability to seamlessly scale the server infrastructure to handle peak usage periods and accommodate sudden surges in demand. This ensures not only a responsive but also an efficiently performing system even under varying workloads.

In conclusion, Django stands out as the preeminent choice for constructing the server infrastructure for our digital document management system. Its modularity, user-friendliness, and scalability collectively address and overcome the challenges posed by the project. Django not only provides the requisite flexibility, customization, and resilience but also ensures ease of use and the seamless scalability demanded by the dynamic nature of digital document management. Harnessing Django's capabilities, we are poised to create a robust and efficient solution that impeccably meets the critical requirements of our project.

3.2.3 Database management system

Considering the goal of our document management system (DMS), choosing the right database engine is extremely important. In our system, the database is expected to be:

Consistency and Relatability:

- Given that our DMS aims for seamless organization and retrieval of documents, the choice of SQL aligns perfectly with the need for data consistency.
- SQL's relational model ensures that data relationships are maintained, reflecting the interconnections between various entities within the system.

Structured Approach to Document Management:

- The structured nature of SQL databases harmonizes with the structured organization of documents in our system.
- This feature facilitates a systematic approach to document storage, retrieval, and categorization, enhancing overall system efficiency.

ACID Compliance for Transactional Integrity:

- SQL databases adhere to the ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring transactional integrity in the face of system failures or interruptions.
- This ensures that each operation on document data is reliably processed, enhancing the trustworthiness of the document management process.

Given the considerations outlined above, our team decided to choose SQL database over non-SQL database, the next step involves a careful evaluation to determine which SQL database engine best suits the unique requirements of our project.

a. MySQL

MySQL [16] is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It is widely used in various applications, from small-scale projects to large-scale enterprises.

Pros:

- Excellent performance for read-heavy workloads.
- Wide community support and a large ecosystem of tools.

- Simple and straightforward to set up and use.

Cons:

- Compared to some other SQL databases, MySQL may have fewer advanced features such as full-text search or JSON-support
- Its ability of handling of complex transactions is a concern.

While MySQL is a robust choice for various applications, including its widespread adoption and user-friendly nature, for our specific document management system, the need for advanced features and strong support for complex queries is a huge concern

b. Microsoft SQL Server

Microsoft SQL Server [17] is a comprehensive relational database management system developed by Microsoft. It is known for its integration with Microsoft's ecosystem and offers a range of features for enterprise-level applications.

Pros:

- Seamless integration with other Microsoft products and technologies.
- Strong support for business intelligence and reporting tools.
- Excellent performance, especially in Windows environments.

Cons:

- Licensing costs can be relatively high.
- Limited cross-platform compatibility compared to some open-source alternatives

Although Microsoft SQL Server integrates seamlessly with the Microsoft ecosystem and offers user-friendly management tools, the associated licensing costs and platform dependence on Windows environments make it less suitable for our project

c. PostgreSQL

PostgreSQL [18], often referred to as Postgres, is an open-source object-relational database system known for its extensibility and standards compliance. It is recognized for its advanced features and support for complex queries.

Pros:

- High level of extensibility with support for custom functions and indexing methods.
- Robust support for complex queries and transactions.
- Conformance to SQL standards and ACID compliance.

Cons:

- May require more resources compared to other databases, impacting performance.

Given the advanced features, extensibility, and strong adherence to ACID principles, PostgreSQL emerges as the optimal choice for our document management system. Its scalability, robust performance, and flexibility make it well-suited to handle the complexities of document storage and retrieval in our project.

3.2.4 Ontology storage

Graph database management systems (GDBMS) are specifically designed to handle and query highly interconnected data. Unlike traditional relational databases that use tables, rows, and columns, GDBMS use graph structures with nodes, edges, and properties to represent and store data. This makes them particularly well-suited for applications that involve complex relationships, such as social networks, recommendation systems, and natural language processing tasks like those involving ontologies. This section compares several GDBMS, including Neo4j, ArangoDB, OrientDB, and Amazon Neptune, and concludes with a focus on the suitability of Neo4j for storing the ontology.

Neo4j

Neo4j is one of the most popular and widely-used graph databases. It is known for its robust performance, high scalability, and intuitive query language, Cypher. Neo4j's architecture is optimized for fast traversal of graph structures, making it ideal for applications that require deep and frequent relationship queries. Key features of Neo4j include:

- **ACID Compliance**: Ensures reliable transaction processing and data integrity.
- **Cypher Query Language**: Ensures reliable transaction processing and data integrity.
- **High Availability and Scalability**: Supports clustering and sharding to handle large datasets and high transaction volumes.
- **Extensive Community and Support**: A large user base and comprehensive documentation enhance its usability and problem-solving resources

ArangoDB

ArangoDB is a multi-model database that supports graph, document, and key-value data models. It uses the ArangoDB Query Language (AQL) for querying, which can handle complex graph traversals and joins across different data models. Key features of ArangoDB include:

- **Multi-Model Flexibility**: Allows users to choose the most appropriate data model for their application.
- **Graph Capabilities**: Provides robust support for graph operations with high performance.
- **Foxx Microservices Framework**: Enables developers to write JavaScript services directly on the database, enhancing functionality and performance.
- **Cluster Support**: Ensures scalability and fault tolerance.

OrientDB

OrientDB is another multi-model database that supports graph, document, object, and key-value models. It is designed to be fast and scalable, with a focus on performance. Key features include:

- **Multi-Model Support**: Allows integration of different data models within the same database.
- **SQL Compatibility**: Supports SQL queries, making it easier for users familiar with SQL to transition.
- **High Performance**: Optimized for fast read and write operations.
- **Distributed Architecture**: Supports horizontal scaling and replication for high availability.

Amazon Neptune

Amazon Neptune is a fully managed graph database service provided by Amazon Web Services (AWS). It supports both Property Graph and RDF graph models, making it versatile for various graph database applications. Key features include:

- **Managed Service**: Eliminates the need for database administration, allowing developers to focus on application development.
- **High Performance and Scalability**: Optimized for handling large-scale graph datasets with low latency.

- **Support for Multiple Graph Models**: Provides flexibility in choosing the graph model that best suits the application.
- **Integration with AWS Ecosystem**: Seamlessly integrates with other AWS services, offering a comprehensive cloud-based solution.

Our choice

While each of the aforementioned graph databases has its strengths, Neo4j stands out as the most suitable choice for storing and managing the ontology for several reasons:

1. **Specialized Graph Database**: Unlike multi-model databases like ArangoDB and OrientDB, Neo4j is purely a graph database. This specialization ensures that all its features and optimizations are tailored specifically for graph operations, resulting in superior performance and efficiency.
2. **Cypher Query Language**: Neo4j's Cypher is highly expressive and intuitive, making it easier to write and understand complex queries. This is particularly beneficial for querying the intricate relationships within WordNet.
3. **Performance and Scalability**: Neo4j's architecture is designed for high-speed traversal and scalability, which is crucial for handling the large and complex graph structure of WordNet.
4. **Robust Community and Support**: The extensive community and comprehensive support resources available for Neo4j facilitate easier troubleshooting, learning, and development.

Neo4j's focus on graph databases, combined with its powerful query language, scalability, and strong community support, makes it the ideal choice for managing ontologies. Its capabilities will not only ensure efficient storage and retrieval of data but also enable sophisticated analysis and querying of the ontology.

3.2.5 Security - Authentication

a. Overview of Json Web Token

JSON Web Token or JWT, is an open standard used to share security information between two parties — a client and a server. Each JWT contains encoded JSON objects, including a set of claims. JWTs are signed using a cryptographic algorithm to ensure that the claims cannot be altered after the token is issued. JSON Web Token (JWT) stands out as a preferred authentication

mechanism over alternatives such as Simple Web Token (SWT) and Session Cookies for several compelling reasons.

JWTs, as an integral part of web security practices, offer a reliable and easily verifiable means of communication between different components within a system. This reliability stems from the cryptographic signature that authenticates the information contained within the token. The versatility of JWTs is particularly notable, extending across various programming languages and platforms. This cross-platform compatibility makes JWTs a favored choice for secure data exchange in distributed systems, facilitating seamless integration and communication between diverse components.

While alternatives such as Simple Web Token (SWT) and Session Cookies exist, the unique combination of security, reliability, and interoperability positions JWTs as a preferred choice for modern authentication mechanisms. The inherent features of JWTs align with the dynamic requirements of contemporary web security, making them a robust and flexible solution for facilitating secure communication between clients and servers in distributed systems.

b. Json Web Token structure

As in **Figure 3.4**, JSON Web Tokens (JWTs) comprise three integral components: Header, Payload and Signature.

- **Header**: The header, serving as the initial segment of a JWT, encapsulates essential information about the token type and the signing algorithm employed. This critical metadata is Base64Url encoded, typically containing two key components: the token type, invariably denoted as ”JWT,” and the signing algorithm, which could be ”HS256” for HMAC with SHA-256 or ”RS256” for RSA with SHA-256. This encoded header provides the foundational information needed for subsequent verification and processing.

- **Payload**: The payload, also known as the claims or body, the payload constitutes the substantive portion of the JWT, carrying essential information about the user or the token itself. This segment is a repository for an array of claims or statements, encompassing standard claims like ”iss” (issuer), ”sub” (subject), ”exp” (expiration time), and customizable claims tailored to the specific requirements of your application. The payload undergoes Base64Url encoding, ensuring a standardized format for secure data transmission.

- **Signature**: The signature represents the linchpin of JWT security, serving to verify the token’s integrity and safeguard against tampering. Created by amalgamating the encoded header, encoded payload, and a secret or private key—dependent on the chosen signing algorithm—the signature is the product of cryptographic operations, be it HMAC (Hash-

based Message Authentication Code) or RSA (Rivest–Shamir–Adleman). The resulting signature, appended to the token, acts as a digital seal, allowing recipients to authenticate the origin and trust the untampered content of the JWT.

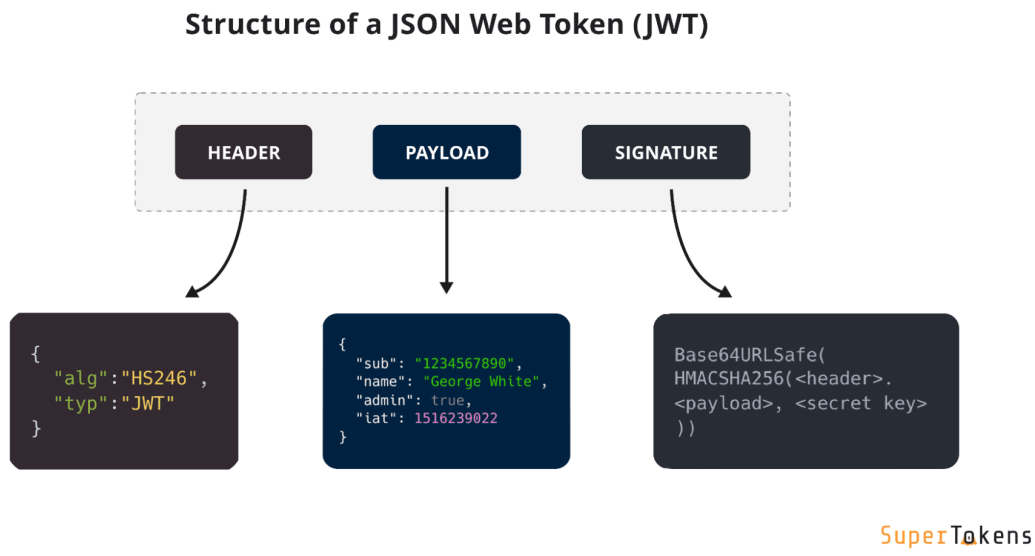

Figure 3.4: Structure of JWT token
Figure reference: https://supertokens.com/blog/what-is-jwt

c. Json Web Token working procedure

To generate a JSON Web Token (JWT), the server undergoes a structured process involving the header and payload. These components are encoded using Base64Url encoding and separated by a period. The server then applies the designated signing algorithm, coupled with a secret or private key, to the encoded header and payload, producing the signature. This resultant signature undergoes additional Base64Url encoding. The final step involves concatenating the encoded header, encoded payload, and encoded signature, punctuated by periods, culminating in the formation of a comprehensive JWT.

When a client sends a request to the server, it typically includes the JWT in the Authorization header, prefixed with the token type (e.g., ”Bearer”). The server receives the JWT and validates it by performing the following steps:

1. It splits the JWT into its three parts: header, payload, and signature.
2. It checks the token’s integrity by recalculating the signature using the received header and payload, along with the secret or public key associated with the signing algorithm.

3. It verifies that the signature matches the one included in the JWT.
4. It checks the standard claims (such as expiration time) in the payload to ensure the token is still valid and within the allowed timeframe.
5. It checks any custom claims specific to the application's requirements, such as user roles or permissions.
6. If all the checks pass, the server considers the JWT valid and proceeds with processing the request. Otherwise, it rejects the token and denies access to the requested resource.

d. Advantages and disadvantages

JSON Web Tokens offer a fundamental advantage through their statelessness. By encapsulating all necessary information within the token itself, JWTs eliminate the need for servers to store session data or conduct frequent database queries for user information. This inherent statelessness not only streamlines the authentication process but also significantly reduces the server's workload. Consequently, this design promotes scalability, allowing for efficient horizontal scaling to handle increased demands without compromising performance.

Additionally, JWTs can carry custom claims, such as user roles or permissions, which can be used for authorization purposes. This customization empowers servers to make fine-grained access control decisions based on user privileges without the necessity of additional database queries. By including these claims directly in the token, the authentication process becomes more efficient, enhancing overall performance and responsiveness. This advantage is particularly valuable in scenarios where tailored authorization is a critical component of system functionality.

However, it's important to consider some potential drawbacks of JWTs. Firstly, since JWTs are self-contained, any information contained within the token is accessible to the client. Therefore, cautions must be exercised to avoid including sensitive data within the payload to prevent unauthorized access. In situations involving confidential information, it is recommended to store such data securely on the server, utilizing the token solely as a reference for retrieval. This careful approach is vital to ensure the integrity and security of sensitive information.

Another consideration is token expiration and revocation. Once a JWT is issued, it remains valid until it expires. This characteristic can pose difficulties in scenarios where a token is compromised or a user's privileges undergo a change. The absence of a direct mechanism for revoking or invalidating a token before its expiration necessitates additional measures. Implementing strategies like refresh tokens or blacklist mechanisms for revocated tokens becomes imperative to manage token validity effectively

Blacklist method

In addressing the complex challenge of efficiently revoking or invalidating JSON Web Tokens (JWTs) before their designated expiration time, one effective strategy involves the implementation of a blacklist method. This method revolves around maintaining a server-side blacklist that keeps track of revoked or invalidated tokens.

Rather than immediately revoking a token when a user's privileges change or when a token is suspected to be compromised, we adopt a more nuanced approach. Instead, the server appends the corresponding token identifier to the blacklist. This blacklist can be seamlessly integrated into various storage solutions, including databases, caches, or any persistent storage mechanism.

This blacklisting process sets the stage for a more proactive approach to token revocation. During the token validation process, the server conducts a diligent check to ascertain if the token identifier exists within the blacklist. If a match is found, indicating that the token has been compromised or invalidated, the server promptly rejects the token. Consequently, access to the requested resource is denied, demonstrating a preemptive and proactive measure to thwart potential security threats.

By adopting the blacklist method, organizations bolster the security posture of their systems, mitigating risks associated with compromised or invalidated tokens. This strategy goes beyond traditional token expiration mechanisms, allowing for real-time responses to changes in user privileges or suspected security breaches. The ability to proactively revoke tokens before their expiration time showcases a sophisticated and forward-thinking approach to token management, contributing to the overall robustness of the authentication and authorization processes within the system. As organizations continue to navigate the evolving landscape of cybersecurity challenges, the blacklist method stands out as a pragmatic solution to fortify the integrity and security of JWT-based authentication systems.

3.2.6 OCR engine

In our project context, we plan to use OCR to extract content from documents, facilitating subsequent full-text searches. The efficiency of the OCR process is crucial, given that each PDF document may comprise over 10 pages. Swift processing is essential to ensure users do not experience prolonged wait times, enabling them to quickly engage with the system.

Consequently, we prioritize an OCR engine with speedy processing capabilities and proficiency in recognizing Vietnamese. While features like handwriting recognition would be advantageous, they are not essential for our primary requirements.

a. Tesseract OCR

Tesseract OCR [19] is an open-source optical character recognition engine known for its transparency and flexibility. Developed by Google, it has a large community supporting ongoing updates. It's designed to convert different types of documents, including scanned paper documents and PDFs, into editable and searchable data. As an open-source project, it encourages community involvement and transparency in its development.

Pros:

- Open-source and freely available.
- Good accuracy for printed text.
- Tesseract supports a multitude of languages including Vietnamese

Cons:

- Tesseract may face accuracy issues, especially with complex layouts or less common fonts.
- Limited performance on handwritten or complex layouts.

b. PaddleOCR

PaddleOCR [20], powered by PaddlePaddle, is an optical character recognition engine that utilizes deep learning techniques. Its focus is on accurately extracting text from images and documents, offering potential advantages in handling complex layouts and diverse fonts.

Pros:

- Offering higher accuracy, especially with complex layouts and various fonts by utilizing deep learning model.
- PaddleOCR may include advanced features like layout analysis

Cons:

- Longer processing time, deep learning models requiring more resources.
- Limited performance for Vietnamese text

c. VietOCR

VietOCR, created by Pham Ba Quoc Cuong, the pre-trained Vietnamese OCR model used in this analysis was downloaded from the VietOCR GitHub page: https://github.com/pbcquoc/vietocr, utilizes a combination of Language models (Seq2Seq and Transformer) and CNN models to address Vietnamese text recognition. The library exhibits promise, showcasing its effectiveness in recognizing text with intricate layouts and diverse fonts, particularly handwritten text.

Pros:

- Good accuracy for Vietnamese handwritten text.
- Good generality, notably high accuracy even on a new dataset.

Cons:

- May require additional fine-tuning.
- Longer processing time, deep learning models requiring more resources.

Considering all the above options and the project's requirements, prioritizing speed and proficiency in recognizing Vietnamese text, Tesseract OCR stands out as the optimal choice. While PaddleOCR and VietOCR offer advanced features like layout analysis and handwritten text recognition, Tesseract's open-source nature, Vietnamese language support, and community-backed updates align perfectly with our need for swift processing of multi-page PDFs. Despite potential challenges with complex layouts, Tesseract's efficiency and versatility make it the preferred OCR engine for our project.

3.2.7 Cloud services

In the dynamic landscape of today's digital era, the integration of cloud services has emerged as a pivotal catalyst, fundamentally transforming traditional document management systems. The adoption of cloud storage within a digital document management system (DMS) brings forth a myriad of advantages, offering a comprehensive solution that caters to the diverse and evolving needs of modern organizations. The decision to leverage cloud services for storing documents in our project is underpinned by a strategic understanding of the multifaceted benefits it brings to the forefront.

One of the standout advantages of cloud storage lies in its ability to facilitate ubiquitous access to documents, transcending geographical constraints. In a world where remote work and distributed teams are increasingly prevalent, the capability to access documents from anywhere

with an internet connection becomes invaluable. Moreover, many cloud services inherently provide real-time collaboration features, empowering multiple users to collaboratively work on the same document concurrently. This not only fosters a sense of teamwork but also significantly enhances overall efficiency in collaborative workflows.

The scalability offered by cloud services is another pivotal attribute that aligns seamlessly with the fluid nature of contemporary organizational requirements. Cloud storage solutions provide organizations with the flexibility to adjust their storage needs dynamically, scaling up or down based on the volume of documents, all without the encumbrance of physical infrastructure limitations. This adaptability proves particularly crucial for businesses experiencing growth or those with fluctuating storage requirements, allowing them to optimize resource utilization efficiently.

Amid the plethora of cloud services available, Amazon S3 stands out as a strategic and ideal choice for our digital document management system project. Amazon S3's unparalleled features, including robust security measures, high durability, and seamless scalability, make it exceptionally well-suited to meet the specific requirements and demands of our ambitious endeavor. By embracing Amazon S3 as our cloud storage solution, we are poised not only to capitalize on the immediate benefits of cloud storage but also to future-proof our document management system against the evolving needs of the digital landscape.

a. Overview about Amazon S3 (Simple Storage Service)

Amazon Simple Storage Service (S3) [21] is a highly scalable and secure object storage service provided by Amazon Web Services (AWS). It serves as a fundamental building block for storing and retrieving any amount of data over the internet. Designed to offer developers and businesses a reliable and durable storage solution, S3 is widely used for various purposes, including data backup, archival, content distribution, and as a storage backend for applications.

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

b. Key features of Amazon S3

Scalability is a pivotal attribute inherent in Amazon S3, seamlessly aligns with the dynamic and ever-expanding nature of modern document management systems. As our digital document repository continues to grow, the flexibility offered by S3 allows us to effortlessly scale storage capacity, ensuring our system's ability to accommodate the evolving volume of documents without any compromise in performance. This scalability not only future-proofs our document management infrastructure but also positions us to meet the demands of increasing data loads and user interactions.

Durability and Reliability stand as cornerstones of Amazon S3's capabilities, assuring the integrity and accessibility of our data. Through the replication of data across multiple geographically dispersed data centers, S3 ensures high durability, safeguarding against potential hardware failures or unforeseen disasters. This redundancy guarantees that our critical documents remain intact and accessible under diverse circumstances, contributing to the robustness of our document management system.

Security. With access controls, encryption options, and identity management tools, S3 ensures comprehensive protection for the confidential documents. This suite of security measures not only guards against unauthorized access but also aligns with industry-standard security protocols, instilling confidence in the integrity and confidentiality of our stored documents.

Versatility is a notable strength of S3, supporting an array of data types, including images, videos, documents, and application data. Its adaptability extends beyond conventional document storage, serving as a storage backend for static website hosting, data lakes, and as a central repository for data storage in cloud-native applications. This versatility positions S3 as a multifaceted solution that caters to diverse data storage needs within our document management system.

Data Management. With features like versioning, lifecycle policies, and object tagging, S3 provides a comprehensive suite of tools. Versioning enables the tracking of changes to objects over time, ensuring a historical record of document modifications. Lifecycle policies enable automatic transitions to lower-cost storage classes or deletion based on predefined rules, facilitating efficient data management strategies.

Global Accessibility is a distinctive feature of S3, as buckets (storage containers) can be created in various AWS regions. This global footprint ensures low-latency performance, enabling users and applications to access S3 buckets from anywhere in the world. This global accessibility enhances user experience and facilitates seamless collaboration among geographically dispersed teams.

Cost-Effective Pricing Model. Last but not least, the cost-effective pricing model of S3 further solidifies its appeal, following a pay-as-you-go structure. Users only pay for the storage they consume and data transfer out of S3. The availability of various storage classes, each with different performance characteristics and costs, empowers users to optimize costs based on their specific requirements. This flexibility in pricing ensures cost-effectiveness and financial efficiency in managing our document storage needs.

3.2.8 Search engine

In the context of the project, our focus is on enhancing document retrieval through the implementation of a full-text search feature. This functionality leverages the content within documents, allowing users to swiftly locate relevant information. By indexing document content and optimizing search processes, we aim to significantly improve user efficiency in navigating through extensive document repositories. This initiative aligns with our broader project goals of creating a user-friendly and efficient document management system, emphasizing quick and precise access to information.

To integrate this feature, we're assessing diverse approaches that align with the project's scope, focusing particularly on solutions capable of efficiently managing extensive volumes of text data given our system's anticipated need to store a significant amount of textual information.

a. PostgreSQL

PostgreSQL [18], as a relational database management system, also offers robust full-text search capabilities through its built-in features. The *tsvector* and *tsquery* data types, along with the *tsvector* functions, enable efficient indexing and searching of text data. PostgreSQL supports various text search functionalities, including stemming, ranking, and phrase search.

Pros:

- Seamless integration with existing PostgreSQL databases.
- No need for a separate search engine, simplifying the architecture.

Cons:

- Limited in terms of scalability compared to dedicated search engines
- Limited performance for large datasets

Given that our system's database is PostgreSQL, its built-in full-text search capabilities provides a solid solution for projects where the full-text search requirements are not the primary

focus, and the database serves multiple purposes. However, for our project, which demands a need for advanced search features, this might not be the appropriate choice, perhaps we need a more dedicated search engine like Elasticsearch.

b. Elasticsearch

Elasticsearch [22] is a widely-used, open-source search engine known for its speed and scalability. It excels in full-text search, offering powerful features like fuzzy matching, relevance scoring, and support for complex queries. It's suitable for handling large volumes of text data and integrates well with various programming languages.

Pros:

- High-speed search capabilities.
- Scalable architecture.
- Rich querying functionalities.
- Support vector-space searching

Cons:

- Resource-intensive for smaller projects
- Requires extra setup and configuration

For various points being made, choosing Elasticsearch as our search engine for the system is optimal due to its reputation for speed, scalability, and advanced full-text search features. With a seamless integration with various programming languages and support for vector-space searching, Elasticsearch aligns well with our project's requirements.

While it may demand additional setup and resources, its robust querying capabilities make it an optimal solution for efficient and powerful full-text search, particularly in handling large volumes of text data.

3.2.9 Message brokers

In optimizing our system for user experience, the integration of a robust message broker is important. Our primary aim is to handle time-intensive tasks like generating vector embeddings for semantic search and running OCR processes, where response times may exceed 10 seconds. Leveraging a message broker allows us to seamlessly manage these computationally intensive operations in the background, ensuring swift execution without imposing delays on users.

By offloading these tasks to the background, the message broker prevents users from experiencing prolonged wait times, aligning perfectly with our commitment to a responsive and user-friendly system. It becomes the key to achieving optimal performance and enhancing the overall user experience.

RabbitMQ

RabbitMQ [23] is a robust message broker that excels in handling complex queuing scenarios. It supports various messaging patterns like pub/sub and point-to-point communication. RabbitMQ ensures reliable message delivery through features such as acknowledgments and persistent queues.

Pros:

- Ensures reliable message delivery with acknowledgments and persistent queues.
- Scales well for handling high message volumes
- Supports various messaging patterns and protocols

Cons:

- Setting up and configuring RabbitMQ can be complex
- Consumes more system resources

Redis

Redis [24], while known primarily as an in-memory data store, also serves as a lightweight and efficient message broker. It supports pub/sub messaging patterns and provides simplicity and speed in message processing.

Pros:

- Easier to set up and configure
- In-memory storage allows for high-speed message processing
- Serves both as a message broker and a data store

Cons:

- Relies on in-memory storage, which might impact durability
- Limited features for complex queuing scenarios

For our document management system, where simplicity and efficiency in message processing are vital, Redis emerges as the preferred choice. Given the nature of our system's messaging requirements, Redis provides a lightweight yet effective solution. Its simplicity in configuration and lower resource consumption aligns well with our project's goals. While RabbitMQ offers robust features, the additional complexity may not be necessary for our specific use case, making Redis a more suitable and streamlined choice for our document management system.

Moreover, Redis can serve a dual purpose by transitioning into a memory cache on the server side. This versatility adds significant value to our document management system, as it allows us to optimize performance by leveraging Redis not only as a message broker but also as a high-speed, in-memory cache

Distributed task queue

To achieve efficient message processing and robust task management in our document management system, we want to utilize Celery and Redis. Redis acts as our message broker, storing and queuing messages for processing. Celery workers will process the task and send the results to database. The proposed architecture can be visualized as **Figure 3.5**

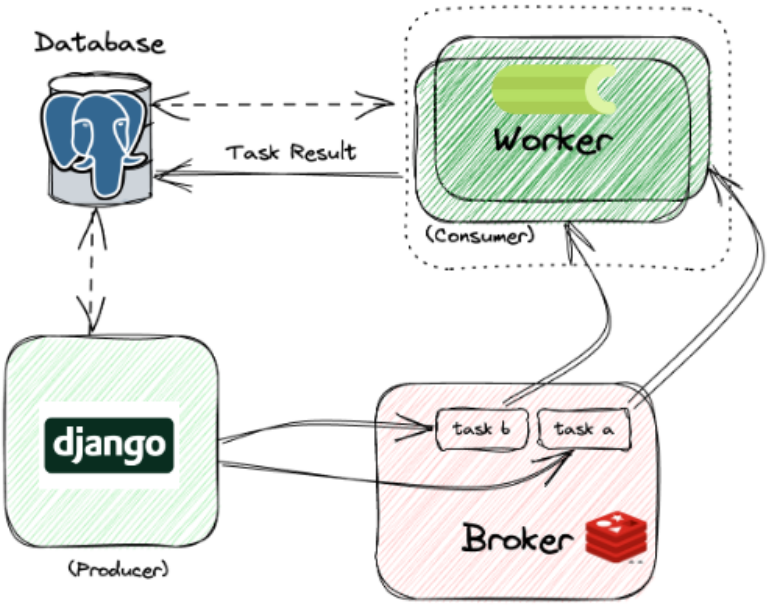

Figure 3.5: Proposed architecture of combining Redis and Celery

Celery

Celery [25] is an open-source distributed task queue system, serving as an efficient asynchronous task manager. It acts as a consumer for distributed task queues, handling the execution of background processes across multiple worker nodes. Widely used in web development

frameworks like Django, Celery enables parallel task execution, making it valuable for managing resource-intensive operations

Given that our server employs Django, the integration of Celery with Redis aligns exceptionally well with the Django ecosystem. Celery, when paired with Redis as the message broker, becomes the consumer that handles the workload efficiently. This combination enhances our system's capabilities by offering distributed task execution, real-time updates, and effective management of background processes. The simplicity of Redis complements Celery's efficiency, making the duo an optimal choice for handling asynchronous tasks, such as document processing and management, within our Django-based document system.

Chapter 4

System analysis and design

In Chapter 3, we focus exclusively on system analysis and design. We investigate stakeholders , clarify both functional and non-functional requirements, and then transition into the investigation and design of essential components. This includes the examination of application schemas, databases, and prototypes, providing a foundational understanding of the project's structure and functionality.

4.1 System analysis

4.1.1 Stakeholders

The primary stakeholders in our digital document management system can be categorized into two main groups: the expert users responsible for creating and editing the ontology and the end-users who will upload and manage documents within the system.

Expert users

Expert users, often subject matter experts or knowledge engineers, are instrumental in the initial stages of system development. Their role involves creating the ontology that forms the backbone of the document management system. During the system analysis phase, it is imperative to engage with these experts to gather insights into the domain-specific terminology, relationships, and classifications relevant to the document management field. Understanding their requirements and ensuring that the ontology accurately represents their knowledge domain lays the foundation for a robust and effective system.

End users

End-users represent the individuals who interact with the document management system on a day-to-day basis. They play pivotal roles in contributing to the efficiency, collaboration, and success of the platform. The system is designed with user-centric features, empowering

individuals across different roles to seamlessly interact with, manage, and benefit from the stored documents.

Stakeholder collaboration

While the end-users take center stage in the system analysis, collaboration with other stakeholders remains crucial. Engaging with any additional stakeholders, such as administrators or IT support personnel, ensures a holistic understanding of the project's requirements. Collaborate with IT support teams to identify technical specifications, integration needs, and potential challenges that may arise during the deployment and maintenance phases.

4.1.2 Functional requirements

Table 4.1: Functional requirements of the system

Requirement ID	Requirement	Description
F01	Authentication	
F01_01	Login	Users can log in to their accounts with proper authentication credentials.
F01_02	Register	Accounts can be registered for new users by admin, providing necessary information for system access.
F01_03	Logout	Users can log out of their accounts, terminating their active sessions.
F01_04	Forgot/Reset password	Provide functionality for users to recover or reset their passwords securely.
F02	User management	
F02_01	View user info	Users can view their personal information, providing transparency and control over their account details.
F02_02	Edit user info	Users can edit their profile information, ensuring accurate and up-to-date user data.
F03	Document Management	
F03_01	Upload Document	Users can upload a document to the workspace of the domain in the system.
F03_02	View Document	Users can view their uploaded documents.

F03_03	Edit Document	Users can edit their documents if they have permission.
F03_04	Remove Document	Users can remove their documents. The removed documents will be moved to the Trash bin.
F04	OCR and Metadata Management	
F04_01	OCR	Utilize Optical Character Recognition (OCR) to extract content from documents.
F04_02	Extract Metadata automatically	Users can submit document and the system extract metadata from the content.
F04_03	Add Metadata	Enhance documents by adding metadata dynamically. User can define metadata of document based on the extracted by system or add more self-defined metadata
F04_04	Edit Metadata	Users can edit and remove the current metadata value of any document if they have permission.
F05	Search	
F05_01	Full-text search	Implement search functionality for documents, facilitating quick retrieval of relevant information.
F05_02	Advanced Search	Provide advanced search options, including filters and parameters, for precise and refined document searches.
F05_03	Query Expansion Search	Implement search functionalities that expand user query with related keywords (defined in the domain ontology)
F05_04	Semantic Search	Search engine capabilities, which includes understanding words from the searcher's intent and their search context
F06	Authorization	

F06_01	Grant permission for documents	Owner users can provide permissions to other users. There are 2 permissions: • View: Users can view documents within their authorized workspaces. • Edit: Authorized users can edit documents, metadata, and other relevant content.
F06_02	Remove Permissions	Owner users can also remove other users' permissions from their documents.
F06_03	Lock File	Implement file locking mechanisms to prevent concurrent editing of the same document by multiple users.
F06_04	Private Setting	Users can set the document to be private or public to other users.
F07	Version Control	
F07_01	Create Version Control	When a document or its metadata is changed. After submitting the changes, a version control is created as the previous version before the updates.
F07_02	Retrieve Version	Users can retrieve any version of the chosen document.
F08	Log Work	
F08_01	User Activities Log	Maintain a comprehensive log of user activities, allowing users to monitor system interactions and changes.
F09	Ontology Management	
F09_01	Create synsets	Create synset and describe its definition.
F09_02	Create words	Add words and add them to their synsets
F09_03	Create relationships between synsets	Define hypernyms and hyponyms of each synsets by creating relationships between them

4.1.3 Non-functional requirements

Table 4.2: Non-functional requirements of the system

Requirement ID	Requirement	Description
NF01	Usability	
NF01_01	Friendly Interface	The system should have a user-friendly and intuitive interface
NF02	Scalability	
NF02_01	Scalability	The system should be scalable depending on the size of the company.
NF03	Integration	
NF03_01	Integration	The system shall integrate seamlessly with external systems, such as email for notification purposes and Amazon S3 for document storage.
NF04	Training requirements	
NF04_01	Trainings requirements	Training sessions shall be conducted for expert users and users to ensure effective system utilization and knowledge of advanced features.
NF05	Performances	
NF05_01	Availability	The system can be accessed at any time, except during scheduled maintenance windows.
NF05_02	Workflow performances	The system should be fast and efficient in document uploading and searching. Not over 5 minutes.
NF05_03	Responsiveness	The system's user interface shall be responsive and accessible on various devices, including smartphones, tablets and PCs.
NF06	Security	
NF06_01	Accessibility	The system can be accessed with the registered username/email and password and denied access when any of them is incorrect.
NF06_02	Personal Information Security	The system should securely store users' personal infomation and access tokens every single login time.

4.2 Use case design

4.2.1 Use case diagram

General use case view

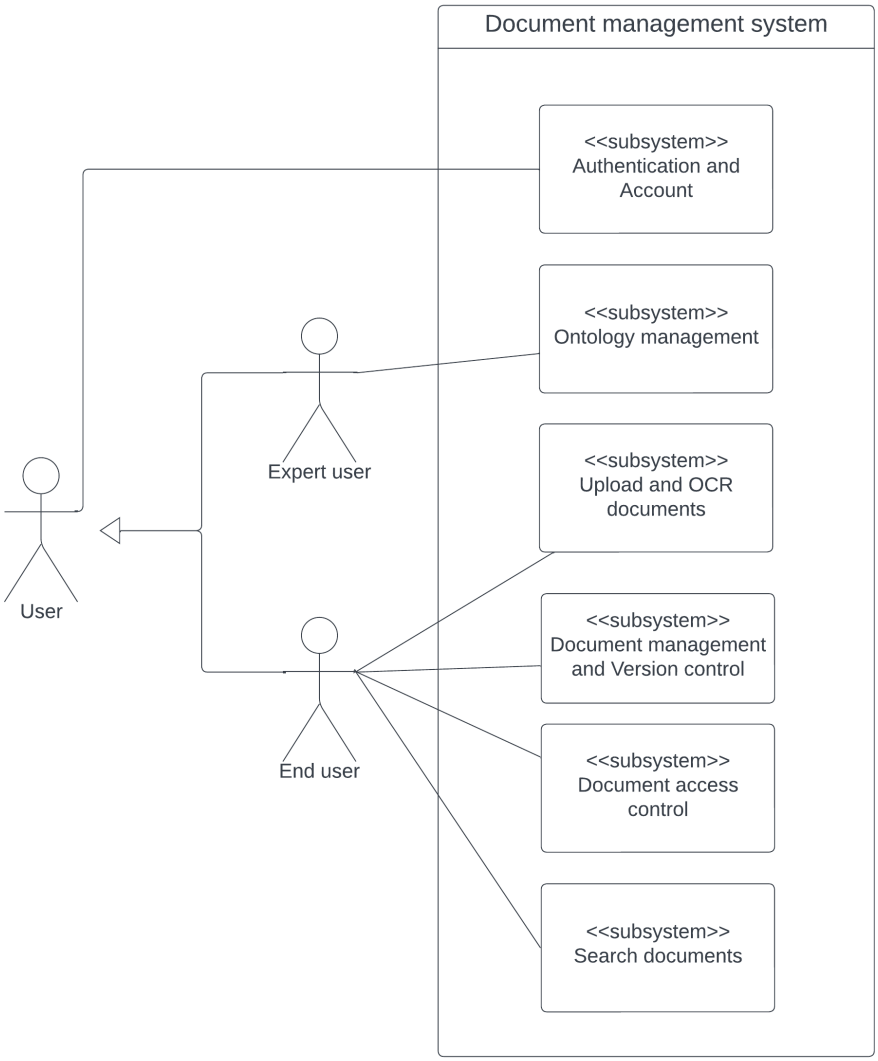

Figure 4.1: General use case view of the system

Figure 4.1 is the general use case overview of the system, which consists of 6 subsystem:

- **Authentication and Account**: This subsystem contains use cases related to authentication, authorization and account management of users
- **Ontology management**: This subsystem contains use cases related to the process of building and modifying ontoloies, which is performed by Expert users

- **Upload and OCR documents**: This subsystem contains use cases related to the process of uploading documents, extracting metadata and saving documents to database and cloud service. Actors participating in this subsystem are end users.
- **Search documents**: This subsystem contains use cases related to the function of searching for documents using different methods. Actors participating in this subsystem are end users.
- **Document management and Version control**: This subsystems contains use cases related to the process of updating document's metadata of uploading a new file, thereby creating a new version of the document.
- **Document access control**: This subsystems contains use cases related to some business processes such as sharing documents and tracking user actions on a document.

Subsystem: Authentication and Account

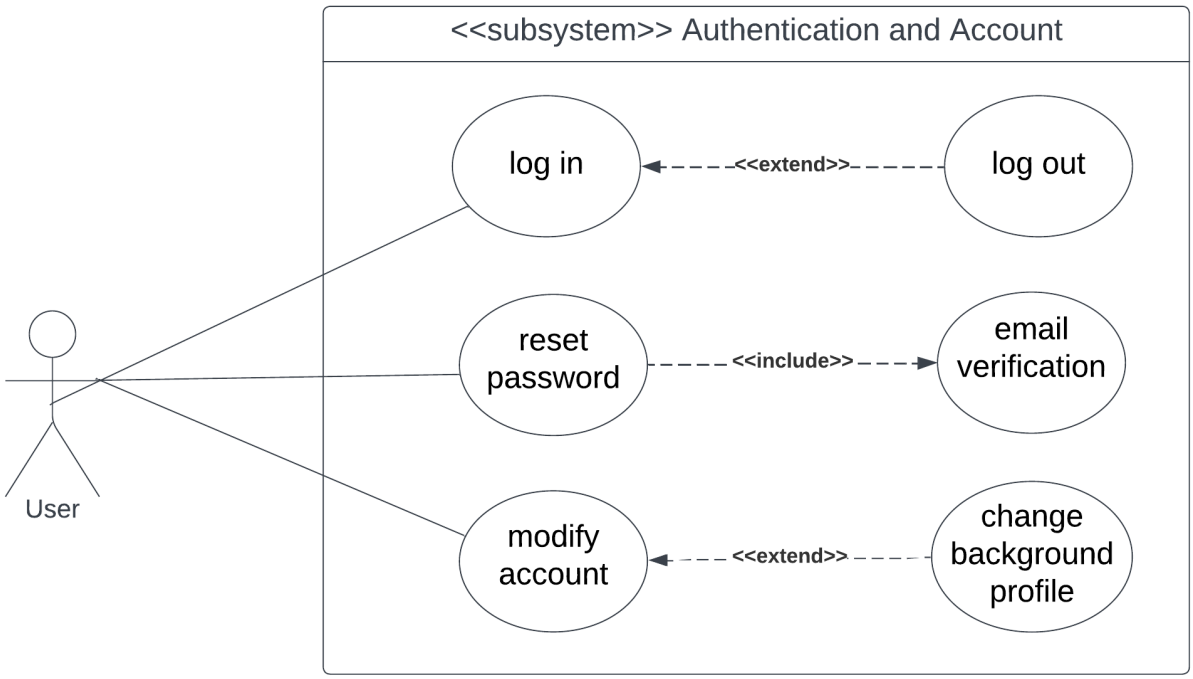

Figure 4.2: Subsystem: Authentication and Account

In the subsystem of **Figure 4.2**, user can perform authentication action such as logging in and resetting password via email verification. User can also modify their account information like updating their background profile. Since this is an application built for internal business, user can not register a new account by themselves, instead, their accounts are provided by the system administrator.

Subsystem: Ontology management

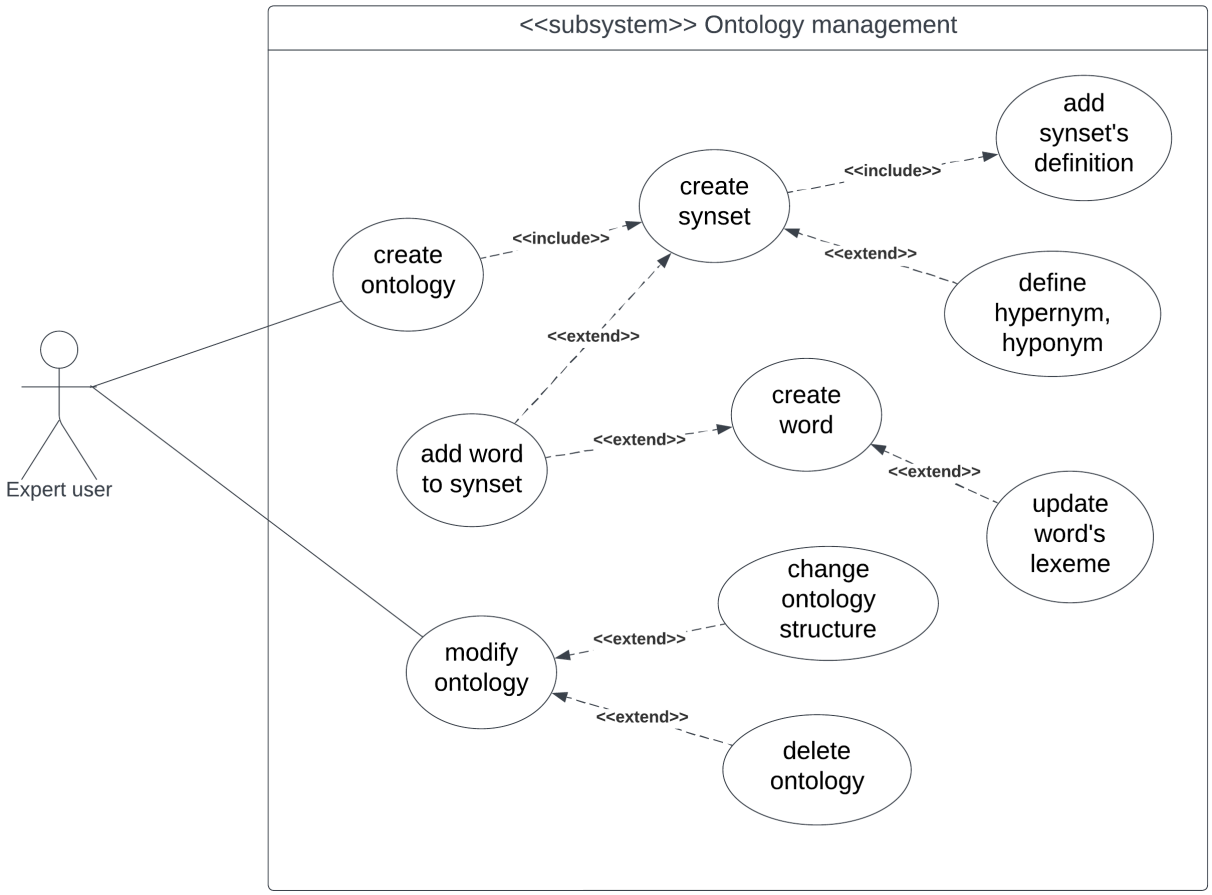

Figure 4.3: Subsystem: Ontology management

In the subsystem of **Figure 4.3**, expert user can create new ontology in their domain expertise. When creating new ontology, user can name the ontology representing its domain. Nodes - which has two types: Synset and Word - are the indispensable parts that form the structure of the ontology. When creating a new Synset, user can give a definition that briefly describe the concept of that Synset and create connection (hypernym/hyponym) between different synsets. When creating a new Word, user can determine which Synset to which that Word belongs. User can also delete a node if they think that it is no more suitable for the ontology structure. When user deletes a node, the system will also delete all connection linked to that node. Finally, user can always came back to modify the ontologies they created to improve its performance. While modifying an ontology, user can perform any actions on the ontology just like when they created it.

Subsystem: Upload and OCR documents

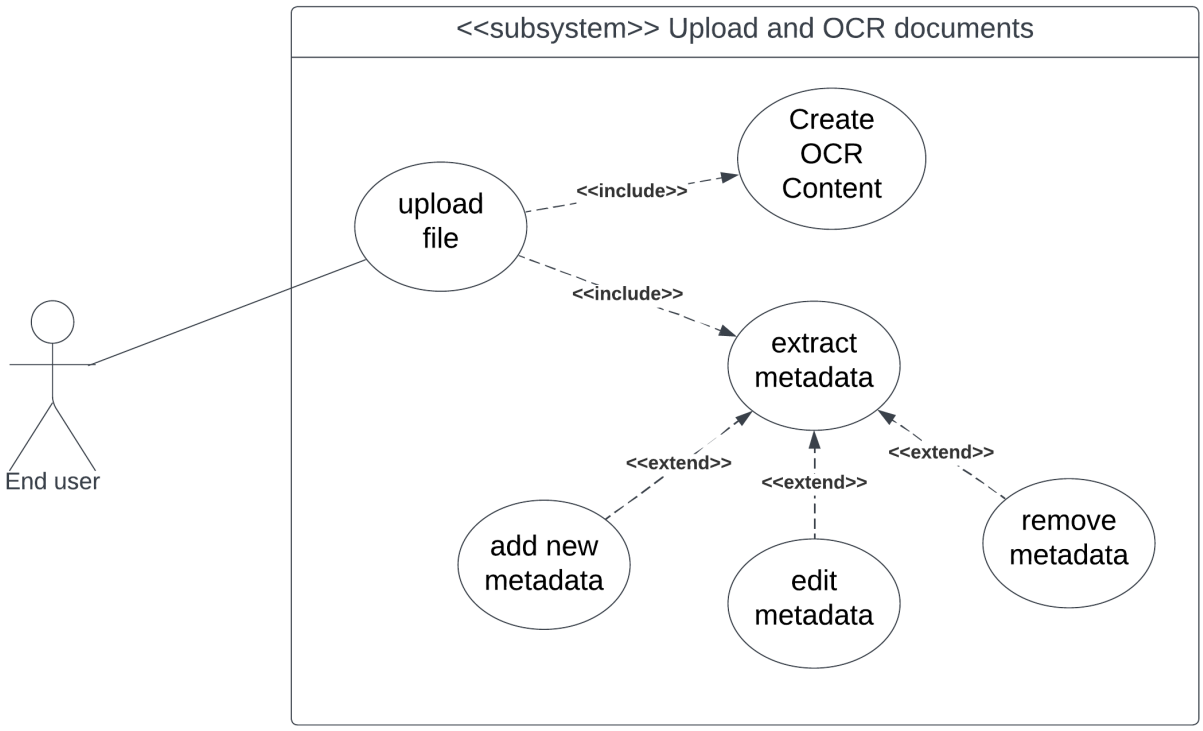

Figure 4.4: Subsystem: Upload and Search documents

In the subsystem of **Figure 4.4**, user can upload a document and extract metadata from it. When metadata are extracted, user can freely edit the metadata set of that document such as creating a new pair of key – value, or modifying the existing metadata extracted by the system. When user finish modifying metadata, the system will store their file in cloud service Amazon S3. User can search for their documents by using four methods that the system offers: full – text search, advanced search, semantic search and expand query using ontology.

Subsystem: Search documents

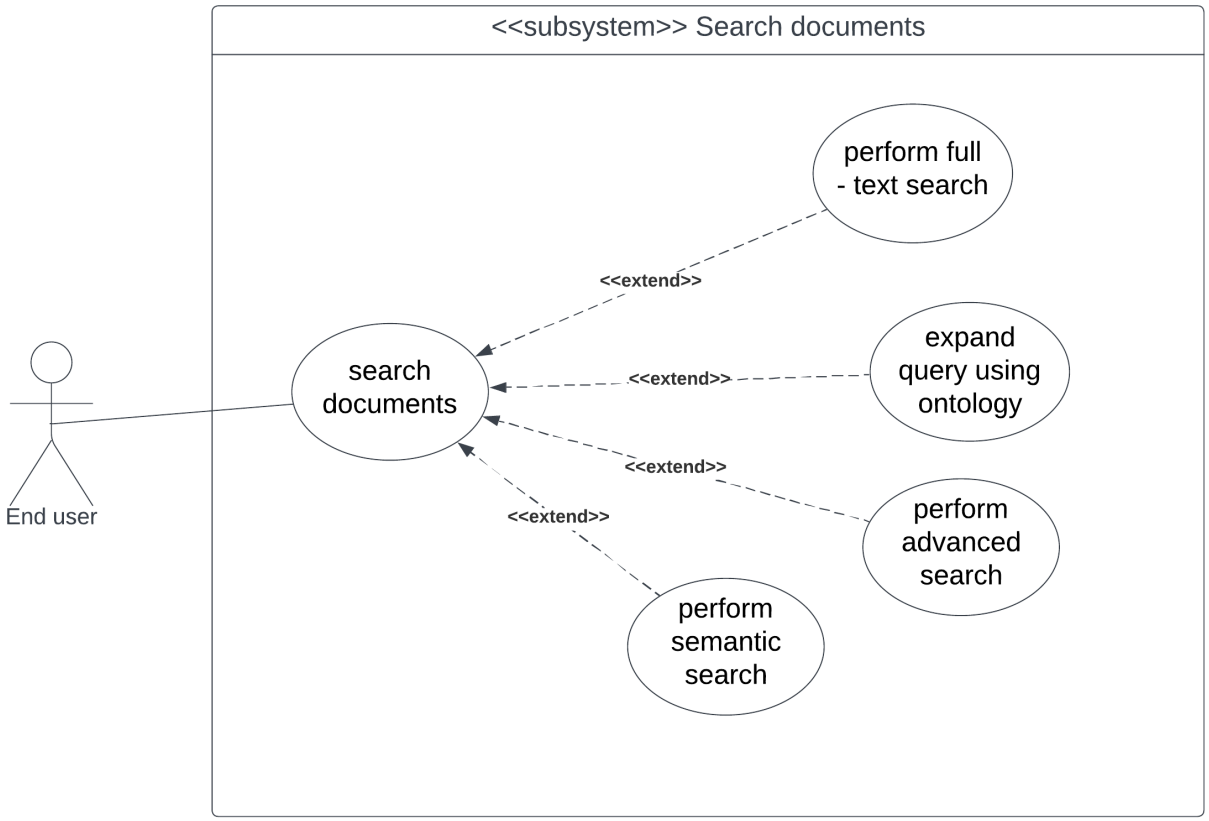

Figure 4.5: Subsystem: Search documents

In the subsystem of **Figure 4.5**, use can search for their documents by using four methods that the system offers: full – text search, advanced search, semantic search and expand query using ontology.

Subsystem: Document management and Version control

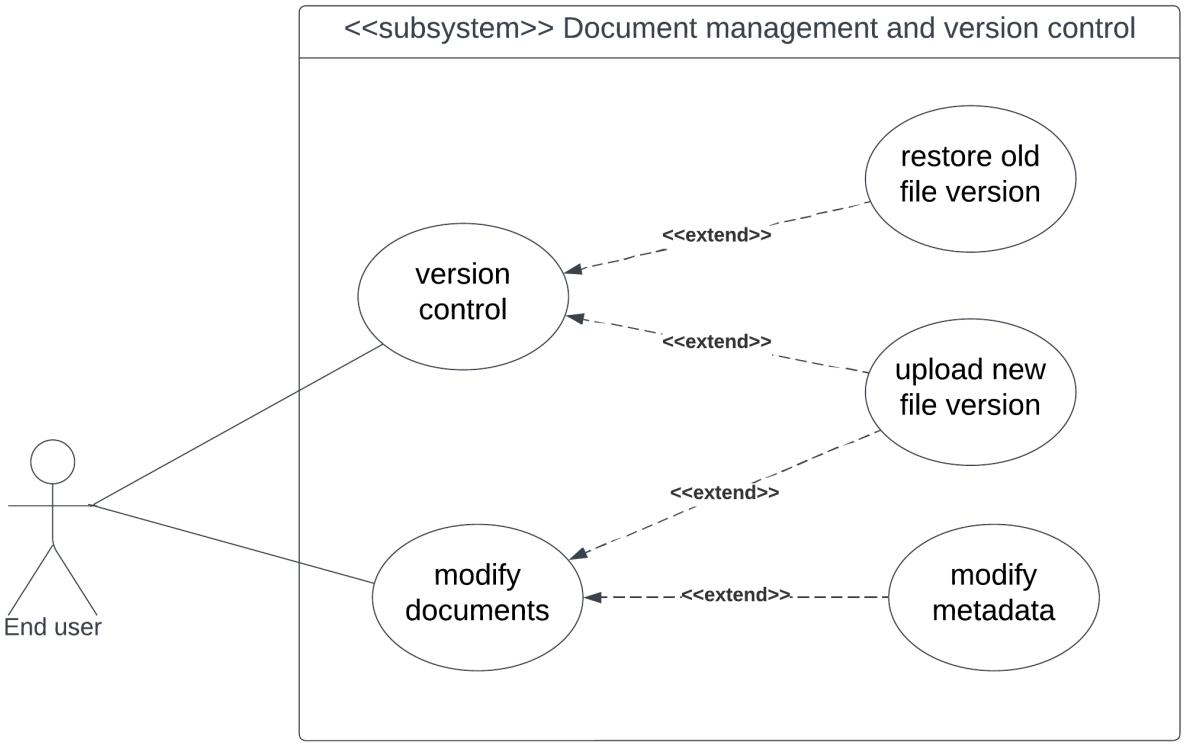

Figure 4.6: Subsystem: Document management and Version control

In the subsystem of **Figure 4.6**, user can manange all versions of their documents. A new version is created whenever user upload a new file version of the document, or when the metadata set of the document is being modified. When a new version is created, the system will record information about the change, such as who made the change, what changes were made, time of the change and the comment of the user who made the change. User can always restore the old version if they want.

Subsystem: Document access control

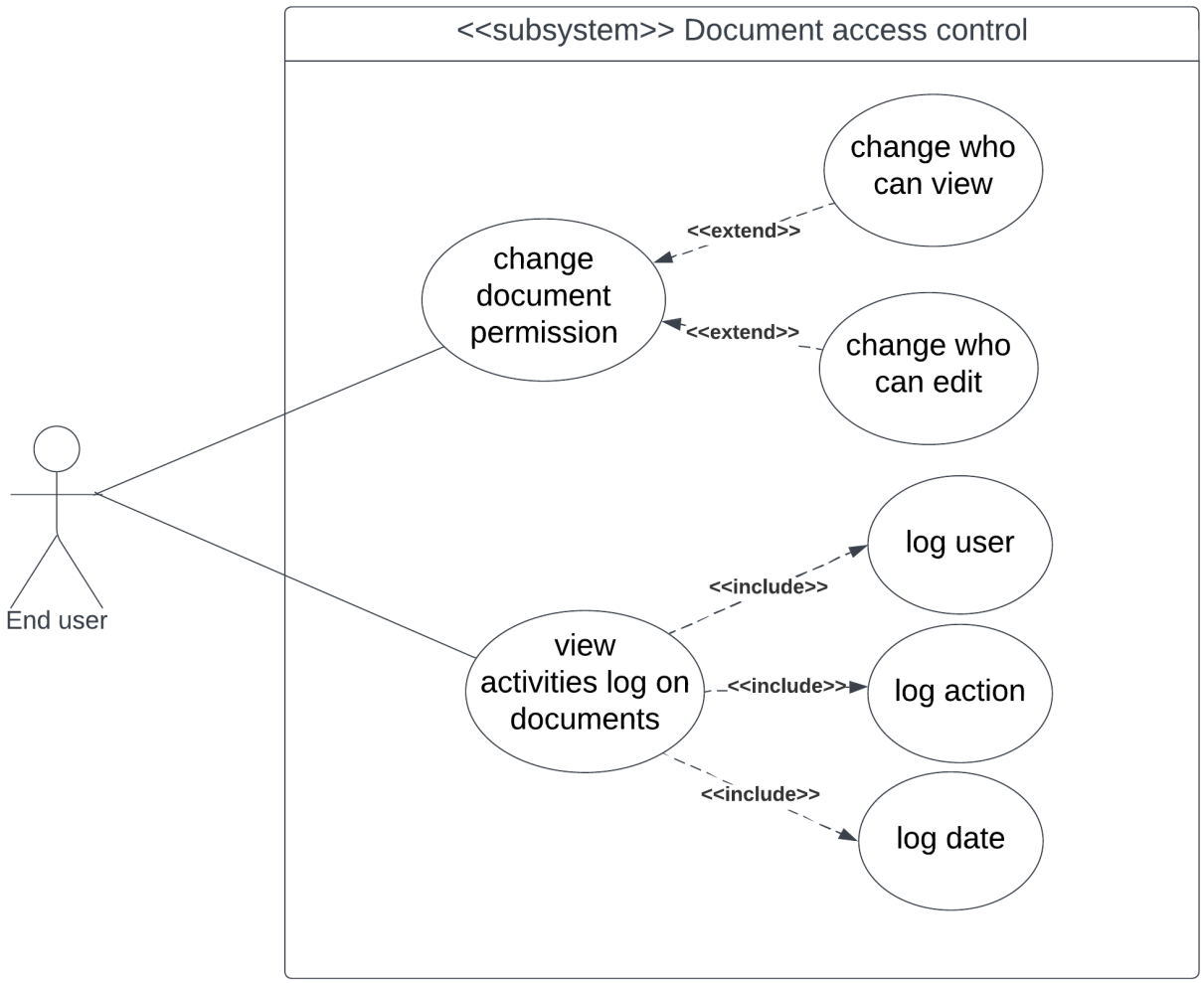

Figure 4.7: Subsystem: Document access control

The last subsystem is related to some business processes on the documents, as illustrated in **Figure 4.7**. User can share their documents to other users by controlling the permissions: who can view and who can edit the file. Finally, all users' action on a document, such as downloading documents, modifying documents,… is recorded. By doing this way, the business can always track down users' activities, thereby preventing security risks.

4.2.2 Use case description

a. Login

Table 4.3: Use case description – Login

Use case ID	01

Name	Login
Description	This use-case describes the process of a user logging into the system
Actor	Users
Trigger	User accesses the system's login page, fills the necessary input field and presses the Login button.
Pre-condition	The user has a registered account.
Post-condition	Upon successful login, the user gains access to the system.
Main flow	1. The user navigates to the system's login page. 2. The system presents the user with input fields for entering their email and password. 3. The user enters their valid email and password. 4. The user submits the login form. 5. The system verifies the provided credentials against the stored user data. 6. If the credentials are valid, the system authenticates the user and grants access. 7. The user is redirected to the system's home or dashboard page. 8. The system logs the successful login activity.
Alternative flow	None
Exception flow	At step 7, if the user enters invalid credentials: 7a. The system displays an error message and the user may retry entering the correct credentials.

b. Reset password

Table 4.4: Use case description – Reset password

Use case ID	02
Name	Password forgot and reset
Description	This use-case describes the process of password recovery and re-setting.
Actor	Users
Trigger	The user clicks on the "Forgot Password" link.

Pre-condition	The user has an existing account in the system.
Post-condition	Upon successful password reset, the user gains access to the system.
Main flow	1. The user navigates to the system's login page. 2. The user clicks on the "Forgot Password" link. 3. The system prompts user to enter their registered email/username. 4. The user provides their email and submits the form. 5. The system verifies the email and sends a password reset link to the user's registered email address. 6. The user checks their email for the password reset link. 7. The user clicks on the provided link which redirects them to the password reset page. 8. The user enters a new password and confirms it. 9. The system validates and updates the password for the user. 10. The user is redirected to the login page. 11. The user logs in using their new password.
Alternative flow	None
Exception flow	In step 5, if the input email/username is invalid: 6a. The system displays a message indicating that the email was not found or the request failed. In step 6, if the user receives no password reset email: 7b. The user is advised to check their email address and try the process again.

c. Modify account

Table 4.5: Use case description – Modify account

Use case ID	03
Name	Modify account
Description	This use-case outlines the process of a user changing their personal details within the system.
Actor	Users

Trigger	The user decides to update their personal information within the system.
Pre-condition	The user is logged into their account.
Post-condition	The user's personal details are successfully updated in the system.
Main flow	1. The user is logged into the system. 2. The user navigates to the "Personal Details" section. 3. Within the "Personal Details" section, the user clicks on the "Edit" or "Change Details" option. 4. The system presents a form displaying the user's current personal details (e.g., name, email, etc.). 5. The user modifies the desired fields (e.g., updates name or email). 6. The user submits the updated details. 7. The system validates the changes and updates the user's profile with the new information. 8. The system displays a success notification confirming the successful update of personal details.
Alternative flow	In step 6, if user decides to cancel the personal details change and click the "Cancel" button: 6a. The user clicks on the "Cancel" or "Discard Changes" option. 7a. The system disregards the changes made by the user, maintaining the previous personal details.
Exception flow	In step 7, if user encounters an issue during personal details modification: 8b. The system displays an error message explaining the issue and the user is advised to correct the details and resubmit.

d. Create OCR content

Table 4.6: Use case description – Create OCR content

Use case ID	04
Name	Create OCR content

Description	OCR processing a PDF document to extract text content
Actor	Users
Trigger	User upload a pdf file
Pre-condition	• User is logged in • File is pdf format • File is uploaded successfully
Post-condition	Text content of the document is extracted and stored.
Main flow	1. User uploads a PDF document. 2. System OCR the PDF file. 3. Text content of the document is stored in the system. 4. Pop-up notification or confirmation of successful OCR.
Alternative flow	None
Exception flow	At step 2, OCR process fail because of technical issues 2a. System returns error notification to user. Use case stopped

e. Upload file

Table 4.7: Use case description – Upload file

Use case ID	05
Name	Upload file
Description	Upload a document to system
Actor	Users
Trigger	User click upload button
Pre-condition	User is logged in
Post-condition	File is uploaded to system

Main flow	1. User upload a file 2. System check for duplicated name 3. System upload file to AWS S3 4. System save file's url from S3 service 5. Upload documents to Elasticsearch 6. Pop-up notification or confirmation of successful upload.
Alternative flow	At step 2, if system finds duplicated filename 2a. System requires user to change filename Use case continues from step 3 At step 4, if file format is pdf 4b. System perform OCR to the file and save file content Use case continues from step 5
Exception flow	At step 3, if file fail to upload 3c. System returns error notification to user. Use case stopped

f. Extract metadata

Table 4.8: Use case description – Extract metadata

Use case ID	06
Name	Extract metadata
Description	Extract Metadata automatically based on system template
Actor	Users
Trigger	User click Extract button and choose template
Pre-condition	• User is logged in • File format is pdf • OCR content is available
Post-condition	Extracted metadata is added to document

Main flow	1. User chooses template from the panel 2. System display the extracted metadata 3. User save the extracted data to document
Alternative flow	At step 2, if user want to correct the extracted metadata 2a. User changes the desired metadata field. Use case continues from step 2
Exception flow	None

g. Search document

Table 4.9: Use case description – Search document

Use case ID	07
Name	Search document
Description	Search by name for all document that the user has access to and return a list of documents and number of results
Actor	Users
Trigger	User click Search button
Pre-condition	• User is logged in • Search type checkbox “by name” is checked
Post-condition	Display a paginated list of documents that match the search keyword with number of results
Main flow	1. System searches for documents with names matching the keyword 2. System returns a paginated list of matching documents. 3. The system displays the number of results found.
Alternative flow	None
Exception flow	None

h. Search full-text document

Table 4.10: Use case description – Search full-text document

Use case ID	08
Name	Search full-text document
Description	Search by text content for all document that the user has access to and return a list of documents and number of results
Actor	Users
Trigger	User click Search button
Pre-condition	• User is logged in • Search type checkbox “by content” is checked
Post-condition	Display a paginated list of documents that match the search keyword with number of results
Main flow	1. User enter query 2. System performs searching by document content 3. System returns a paginated list of matching documents. 4. The system displays the number of results found.
Alternative flow	At step 2, user can combine various search methods to enhance the query result 2a. User add other search methods to the current query Use case continues from Step 3
Exception flow	None

i. Perform advanced search

Table 4.11: Use case description – Perform advanced search

Use case ID	09
Name	Perform advanced search
Description	Filtering documents by combining multiple search criteria in an advanced search mode and return a list of satisfied documents and number of results
Actor	Users
Trigger	User click Search button

Pre-condition	• User is logged in • Search type checkbox “Advanced” is checked
Post-condition	Display a paginated list of documents that satisfied all the search keywords with number of results.
Main flow	1. User adds one or more different criteria based on metadata, query conditions, keyword conditions 2. System search documents that satisfy all the specified criteria 3. System returns a paginated list of matching documents.
Alternative flow	At step 1, user want to combine different keywords of the same criteria 1a. User click “+” button next to criteria Use case continues at step 1
Exception flow	None

j. Expand search query

Table 4.12: Use case description – Expand search query

Use case ID	10
Name	Expand search query
Description	Expand current query by adding synonyms and suggest keyword based on the given ontology
Actor	Users
Trigger	User click Expand query button
Pre-condition	User is logged in
Post-condition	• Display a paginated list of documents that match the search keyword with number of results • Display relevant keywords to initial query

Main flow	1. User choose the ontology domain from the Ontology list 2. User enter query 3. System parse the query into separate keywords 4. System search for related keywords on the ontology trees. 5. System expands current query with founded keywords. 6. System search for document using the expanded query. 7. System displays the result together with the related keywords from the initial query.
Alternative flow	None
Exception flow	None

k. Modify metadata

Table 4.13: Use case description – Modify metadata

Use case ID	11
Name	Modify metadata
Description	User modifies the metadata of a document
Actor	Users
Trigger	User clicks on button “Update” from the document info page
Pre-condition	User must have permission to edit the document
Post-condition	User successfully update metadata
Main flow	1. User clicks on button “Update”. 2. System navigates user to document update page. 3. User clicks on “+” button to add a field for typing new metadata. User types a pair of key – value and click “Add”. 4. System displays the newly added metadata. 5. User notes a message for the new change they have made. 6. User clicks on “Save”. 7. System creates a new version of the document with the changes that user have made, and record the message that user notes for the change.

Alternative flow	At step 3, user wants to delete a metadata 3a. User clicks on "Trash" button next to a metadata to delete that metadata 4a. System deletes the metadata Use case continues from step 5 At step 3, user wants to modify a metadata 3b. User clicks on the input field of the key/value that they want to modify and type a new key/value 4b. System records the change Use case continues from step 5
Exception flow	None

l. Upload new file version

Table 4.14: Use case description – Upload new file version

Use case ID	12
Name	Upload new file version
Description	User uploads a new version of the file
Actor	Users
Trigger	User clicks on button "Update" from the document info page
Pre-condition	User must have permission to edit the document
Post-condition	User successfully upload new file version

Main flow	1. User clicks on button “Update”. 2. System navigates user to document update page. 3. User clicks on “Upload new file” button and choose the desired file. 4. System starts extracting metadata from the document. 5. User modifies the extracted metadata field. 6. User notes a message for the new change they have made. 7. User clicks on “Save”. 8. System saves the uploaded file to cloud service Amazon S3. 9. System creates a new version of the document with the changes that user have made, and record the message that user notes for the change.
Alternative flow	None
Exception flow	None

m. Restore old version

Table 4.15: Use case description – Restore old version

Use case ID	13
Name	Restore old version
Description	User restore an old version of the file
Actor	Users
Trigger	User clicks on button “Restore” next to a version of the file
Pre-condition	User must have permission to edit the document
Post-condition	User successfully restores the old version
Main flow	1. User clicks on button “Restore”. 2. System displays a modal asking user if they really want to restore this version. 3. User clicks on option “Yes” 4. System restores the old version of the file.
Alternative flow	None

Exception flow	At step 3, user does not want to restore the old version 3a. User clicks on option “No”. 4a. System closes the modal. Use case continues from step 5

m. Change document permission

Table 4.16: Use case description – Change document permission

Use case ID	14
Name	Change document permission
Description	User changes permission to access the document
Actor	Users
Trigger	User clicks on button “Manage access control” on the document info page
Pre-condition	User must be the owner of the document
Post-condition	User successfully updates permission of the document
Main flow	1. User clicks on button “Manage access control”. 2. System displays a modal showing the current access status: who can view and who can edit the document. 3. User enters the username of the person they want to give permission 4. User specifys the access level of this person: can view or can edit 5. User clicks on button “Save” 6. System saves the new permission on the document.

Alternative flow	At step 3, user wants to change access level of a person 3a. User changes the access level through the select form next to the username. 4a. System updates the new access level of the selected person. Use case continues from step 5 At step 3, user wants to delete permission of a person 3b. User click on "Trash" button next to the username. 4b. System deletes the permission of the selected person. Use case continues from step 5
Exception flow	None

n. Create ontology

Table 4.17: Use case description – Create ontology

Use case ID	15
Name	Create ontology
Description	User creates a new ontology for the system
Actor	Expert user
Trigger	User clicks on button "Create new ontology"
Pre-condition	User must log in as expert user
Post-condition	User successfully create new ontology
Main flow	1. User clicks on button "Create new ontology". 2. System displays a modal for entering new ontology name. 3. User enters name of the new node and click "Create". 4. System creates new ontology and navigate user to ontology dashboard page 5. System saves the newly created ontology to the database.
Alternative flow	None
Exception flow	None

o. Create synset for ontology

Table 4.18: Use case description – Create synset for ontology

Use case ID	16
Name	Create synset for ontology
Description	User add a new synset to the ontology
Actor	Expert user
Trigger	User clicks on button “Add synset”
Pre-condition	User must log in as expert user
Post-condition	User successfully add new synset
Main flow	1. User clicks on button “Add synset”. 2. System adds new synset to the ontology. 3. User gives the synset some definition. 4. System saves the synset’s definition and create vector embedding for it. 5. User types the new word in the Sense list of the Synset and click “Add”. 6. System checks for existance of that Word in database, if it already exists, system links that Word to the current Synset. 7. User links this synset as hypernym to another synset by searching for Synset ID in the Children field and click “Add” button 8. System create connection between two synsets 9. Use case iterates repeatedly from step 1 to step 8 base on the user requirement for the ontology.
Alternative flow	At step 7, user wants to links this synset as hyponym to another synset 7a. User search for Synset ID in the Parent field and click “Add” button Use case continues from step 8
Exception flow	None

4.3 Workflow design

4.3.1 Login and forgot password

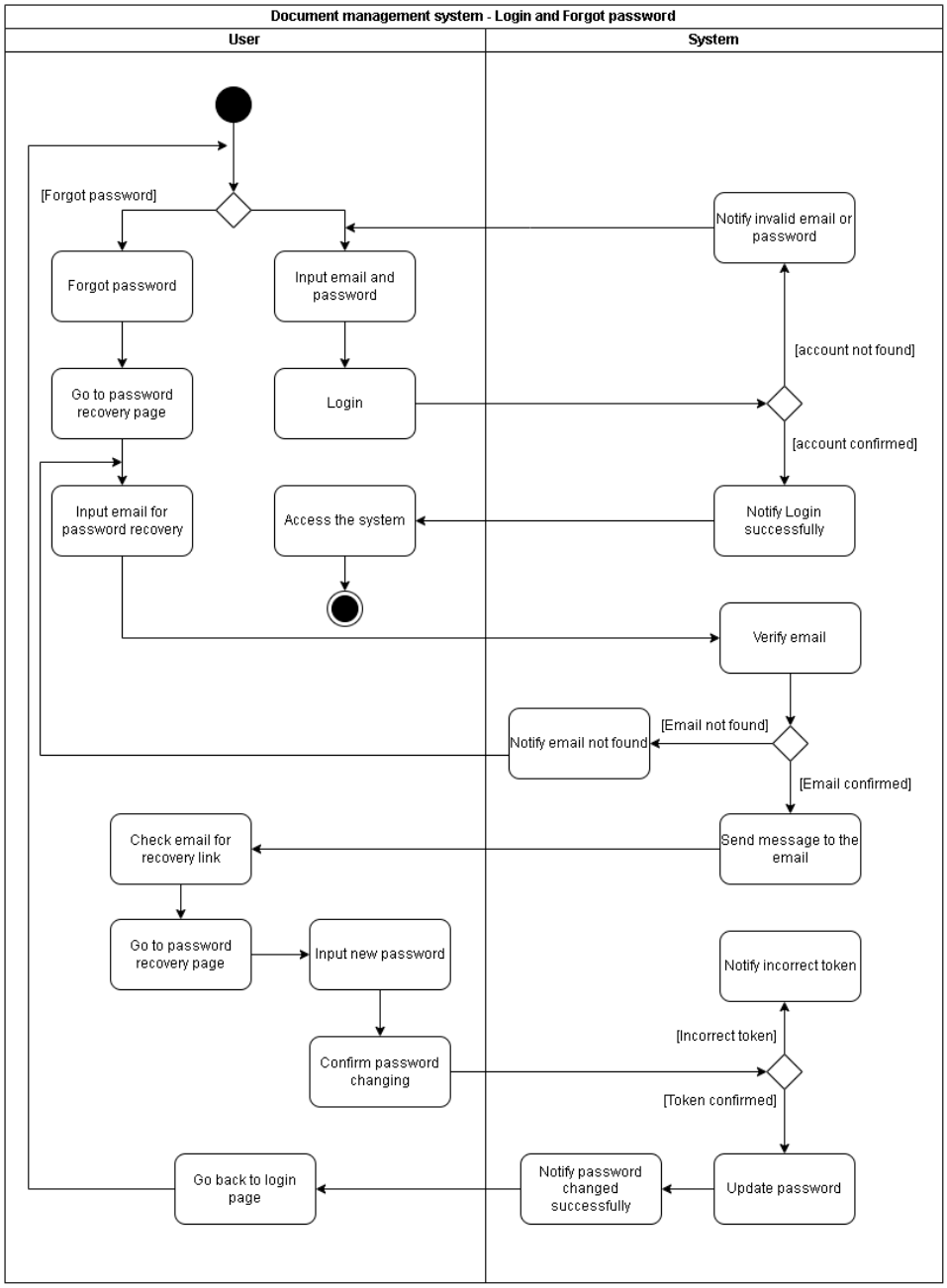

Figure 4.8: Activity diagram – Login and forgot password

As demonstrated in **Figure 4.8**, users are presented with two options: entering their email and password or utilizing the forget password feature.

For the first option, users input their email and password on the login screen. The system

then verifies the existence of the provided email in the database and ensures the correctness of the password. If successful, users gain access; otherwise, they receive a notification indicating an ”Incorrect email or password.”

In the event of forgotten passwords, users select the ”Forgot Password” button, leading them to a password recovery page. Here, users input their recovery email, and the system checks for its presence in the database. If the email is valid, a password recovery email containing a unique token/link is dispatched. In the case of an invalid email, users are notified with an ”Incorrect email” message. Upon receiving the recovery email, users click the link to access the password recovery page. They then enter a new password and submit the form along with the received token. The system verifies the token’s correctness; if valid, the password is updated in the database. Conversely, an ”Incorrect token” message is displayed for an invalid token.

After the “forgotten passwords” process, users are informed of the successful password update, allowing them to log in using their new credentials. This comprehensive approach ensures a secure and efficient password recovery process.

4.3.2 Upload Document

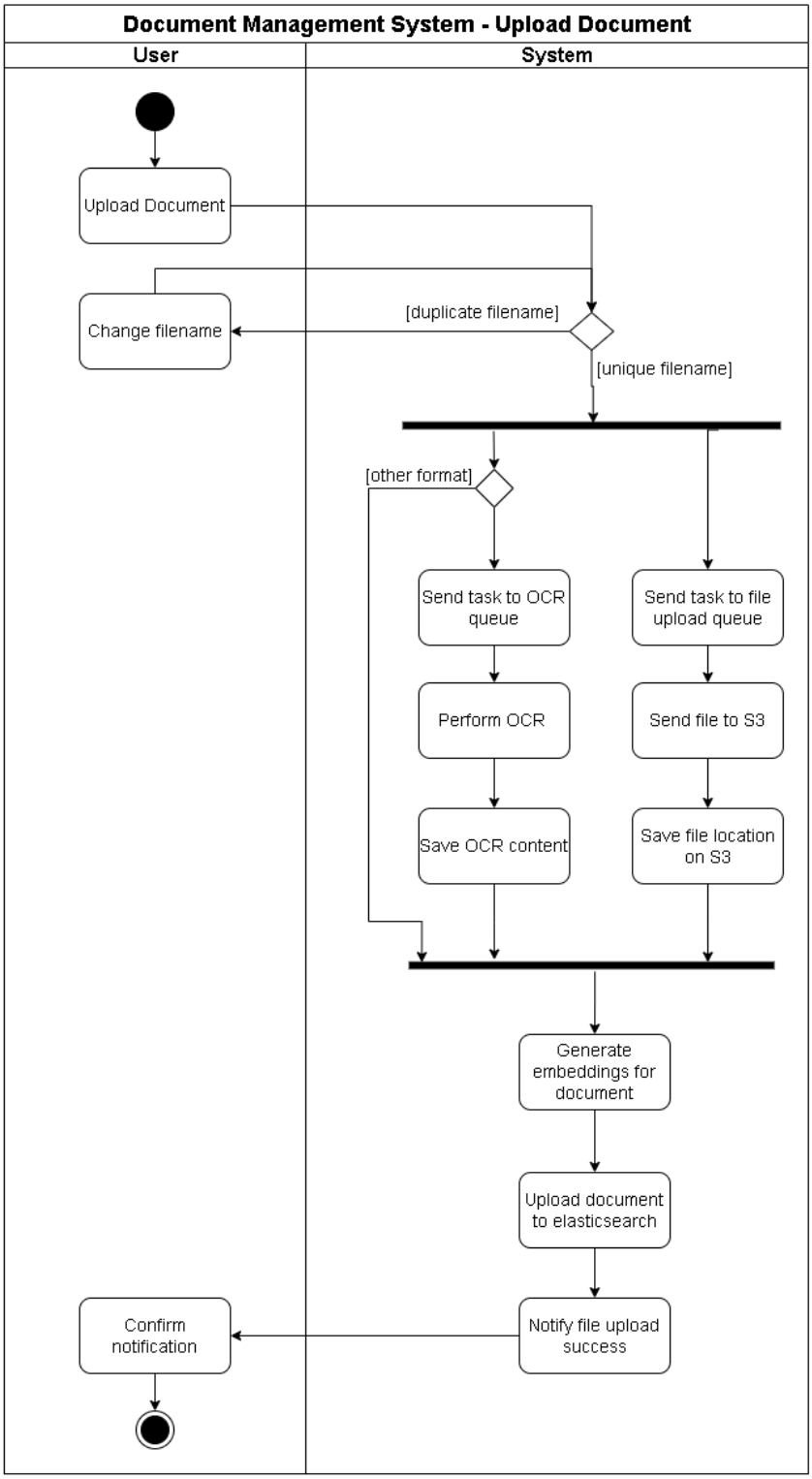

Figure 4.9: Activity diagram – Upload document

Workflow for the ”Upload Document” Activity Diagram in **Figure 4.9**: When a user initiates a file upload, the system first checks for the existence of the file name in the system. In case of duplication, the user is prompted to change the filename, or an automatic renaming occurs after further processing. Subsequently, the system concurrently executes two tasks by dispatching them to the task queue, optimizing response time. The first task involves generating OCR content for PDF files, skipping this process if the file is in a different format. Simultaneously, the second task uploads the file to the S3 cloud storage and saves the retrieval link. Following this, the system generates initial embeddings for the document and uploads both the embeddings and the document to Elasticsearch for indexing. Finally, the user receives a notification regarding the status of the upload process.

4.3.3 Extract Metadata

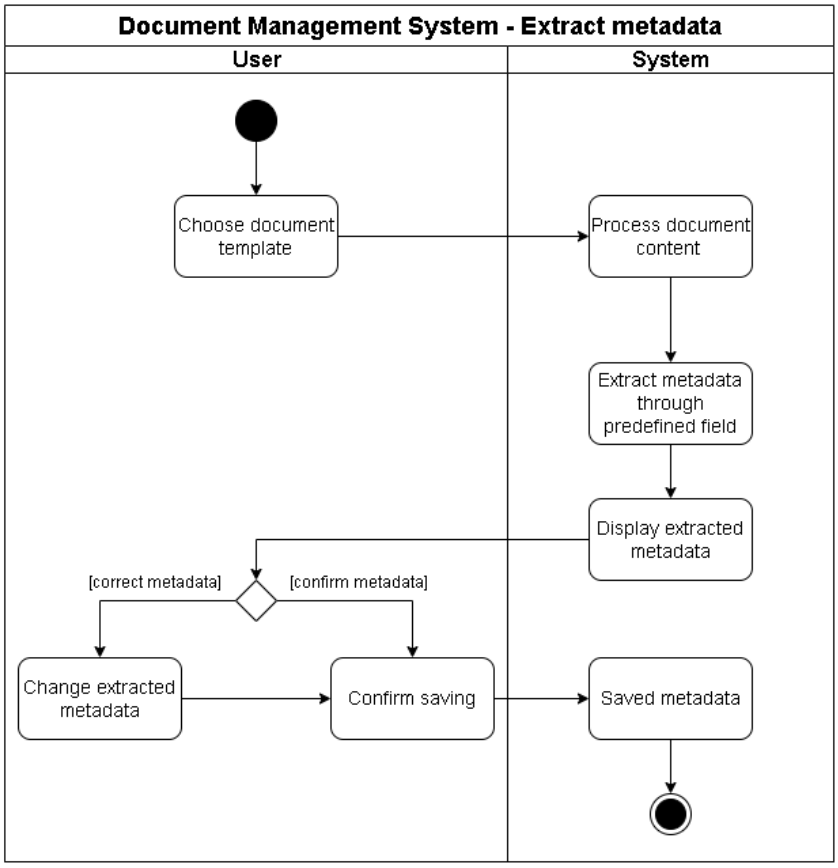

Figure 4.10: Activity diagram – Extract metadata

Workflow for the ”Extract Metadata” Activity Diagram in **Figure 4.10**: When a user seeks to automatically extract document metadata using a provided document template, the process follows in several steps. Initially, the user selects the desired template. Subsequently, the predefined fields from the document template are extracted, and the resulting metadata is displayed

to the user. Following this, the user has the option to modify the extracted metadata if any errors or inaccuracies are identified. Lastly, upon user confirmation, the system saves the updated metadata.

4.3.4 Expand Search Query

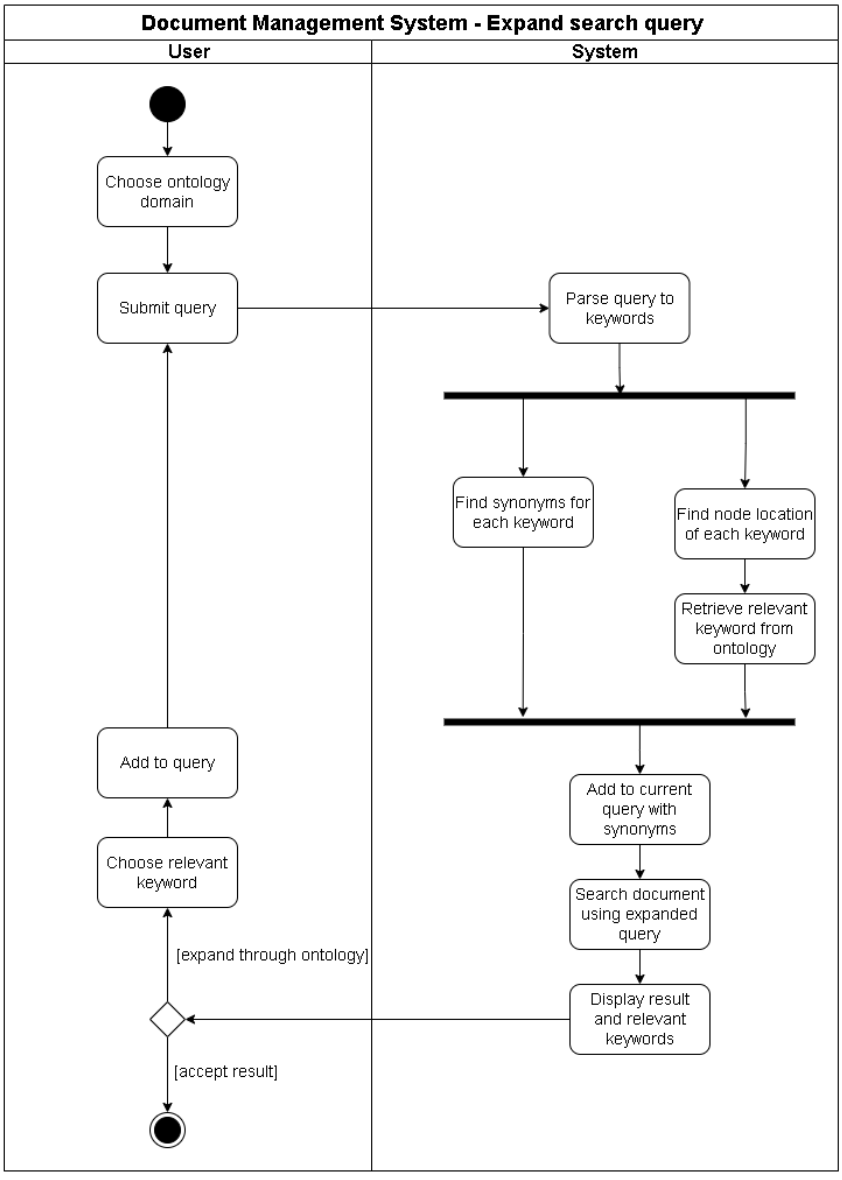

Figure 4.11: Activity diagram – Expand search query

Workflow for the ”Expand Search Query” Activity Diagram in **Figure 4.11**: When a user aims to broaden the current search query, the process involves several key steps. Firstly, the user selects the domain ontology for expansion. The system then parses the query, identifying meaningful keywords and searching for their synonyms. Simultaneously, the system navigates

the ontology tree to retrieve relevant keywords associated with the query. Subsequently, the system incorporates these synonyms into the initial search query, forming an expanded query that is sent to the search engine for results. In parallel with the initial search results, the user has the option to further broaden the query scope by adding relevant keywords extracted from the domain ontology.

4.3.5 Semantic Search

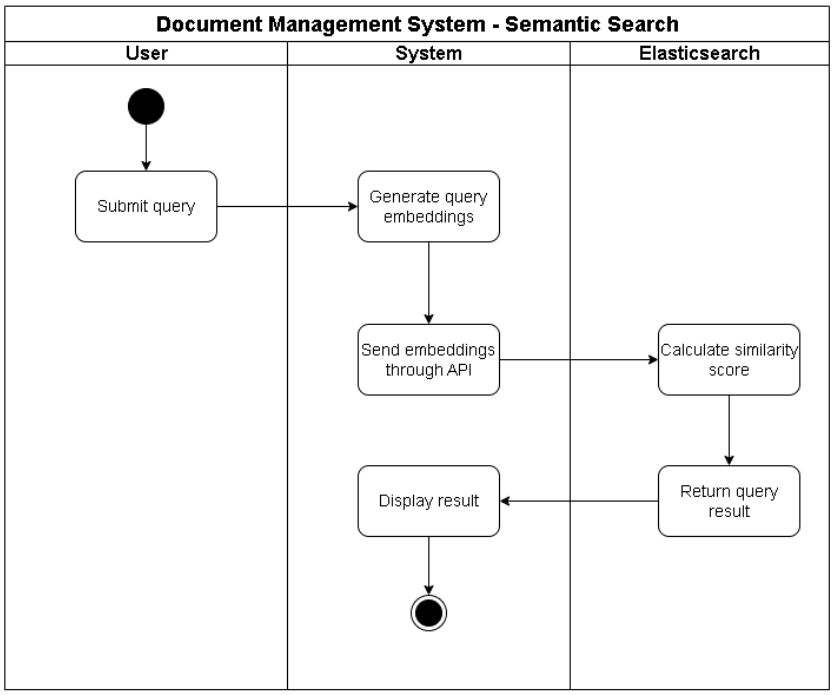

Figure 4.12: Activity diagram – Semantic search

Workflow for the ”Semantic search” Activity Diagram in **Figure 4.12**: The semantic search workflow begins with preprocessing the search query through a tokenizer. Subsequently, the tokenized query is embedded into a 768-dimensional dense vector space using the NLP model. The next step involves calculating semantic similarity between the tokenized query and documents in Elasticsearch. Finally, search results are ranked based on the calculated similarity scores, ensuring a refined and relevant outcome for the user.

4.3.6 Modify Document

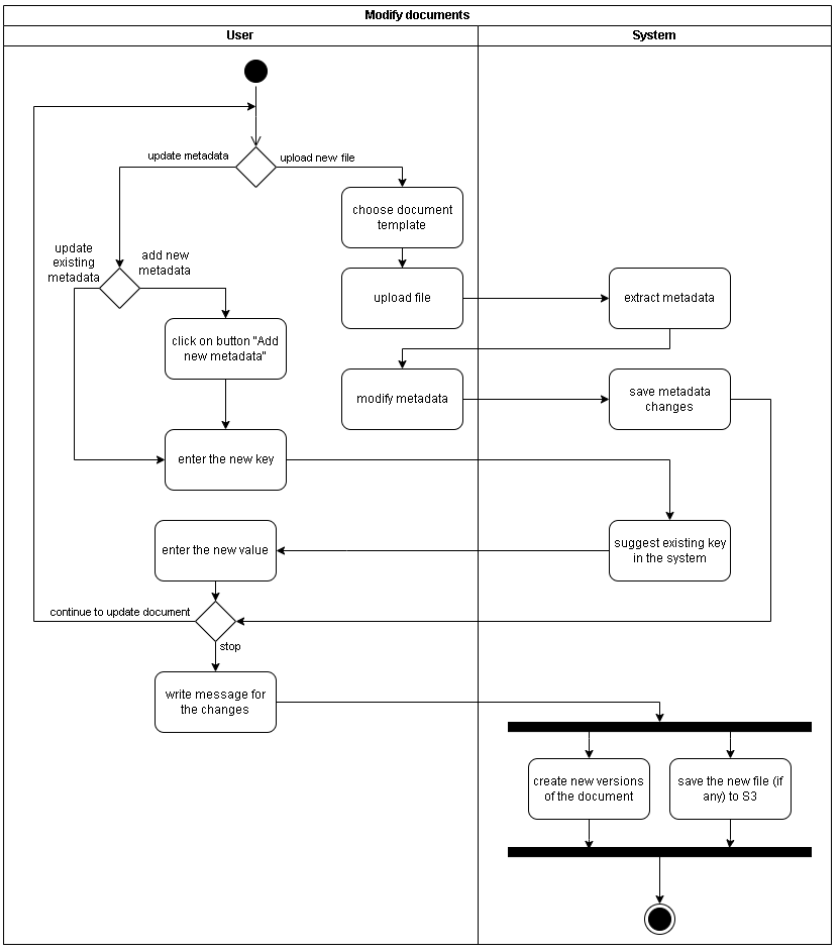

Figure 4.13: Activity diagram – Modify document

In this workflow in **Figure 4.13**, users can upload a new file or modify metadata. Users can upload a new file by choosing the document template, uploading the file and the system will extract metadata from the new file. While adding new metadata, the system will use live search to suggest existing keys related to the user's query. Users can leave messages indicating what they have changed in this document. Finally, the system creates a new version of the document incorporating those changes, and if a new file is uploaded, the system will store that file to Amazon S3.

4.3.7 Change Permission

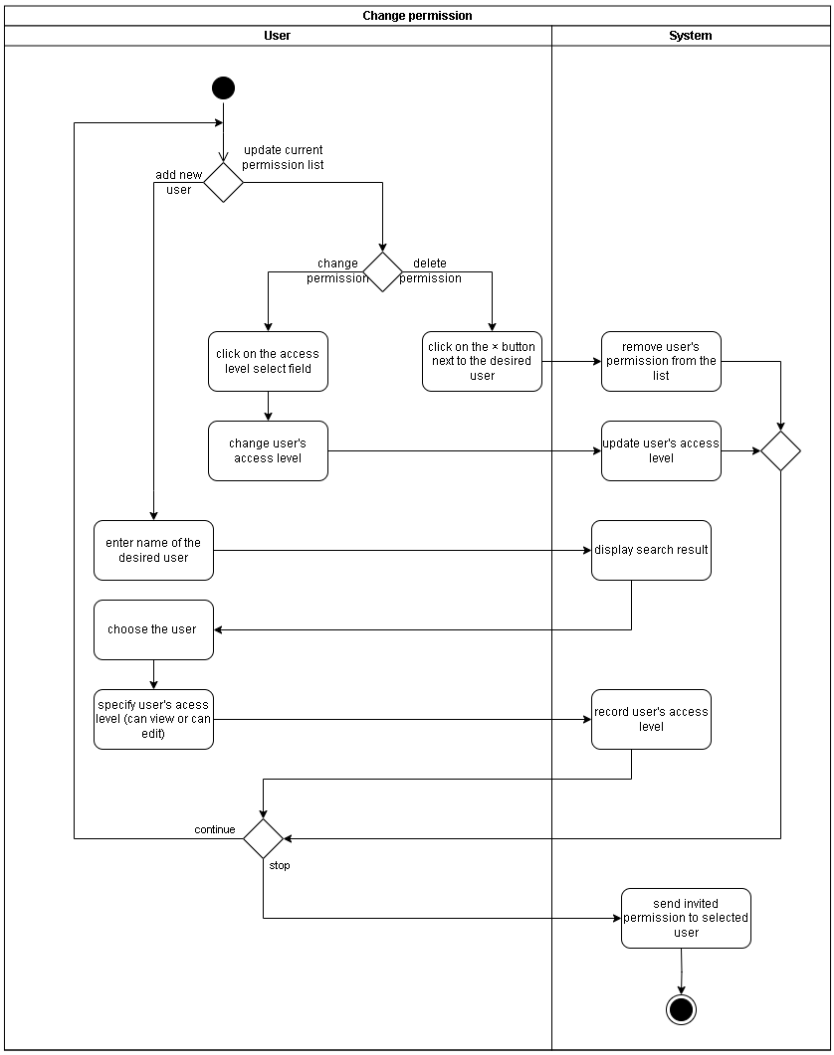

Figure 4.14: Activity diagram – Change permission

In this workflow in **Figure 4.14**, users can add a new user to the access list or update the current list. If users decide to add a new user, they can type their name into the search box and the system will display search results. After choosing the desired person, users continue to choose their access level, whether they can edit or just view the document. Users can also remove other users' access by clicking the "×" button next to their name. When complete, users click "Save" and all changes to the permissions on this document will be updated in the database.

4.3.8 Create Ontology

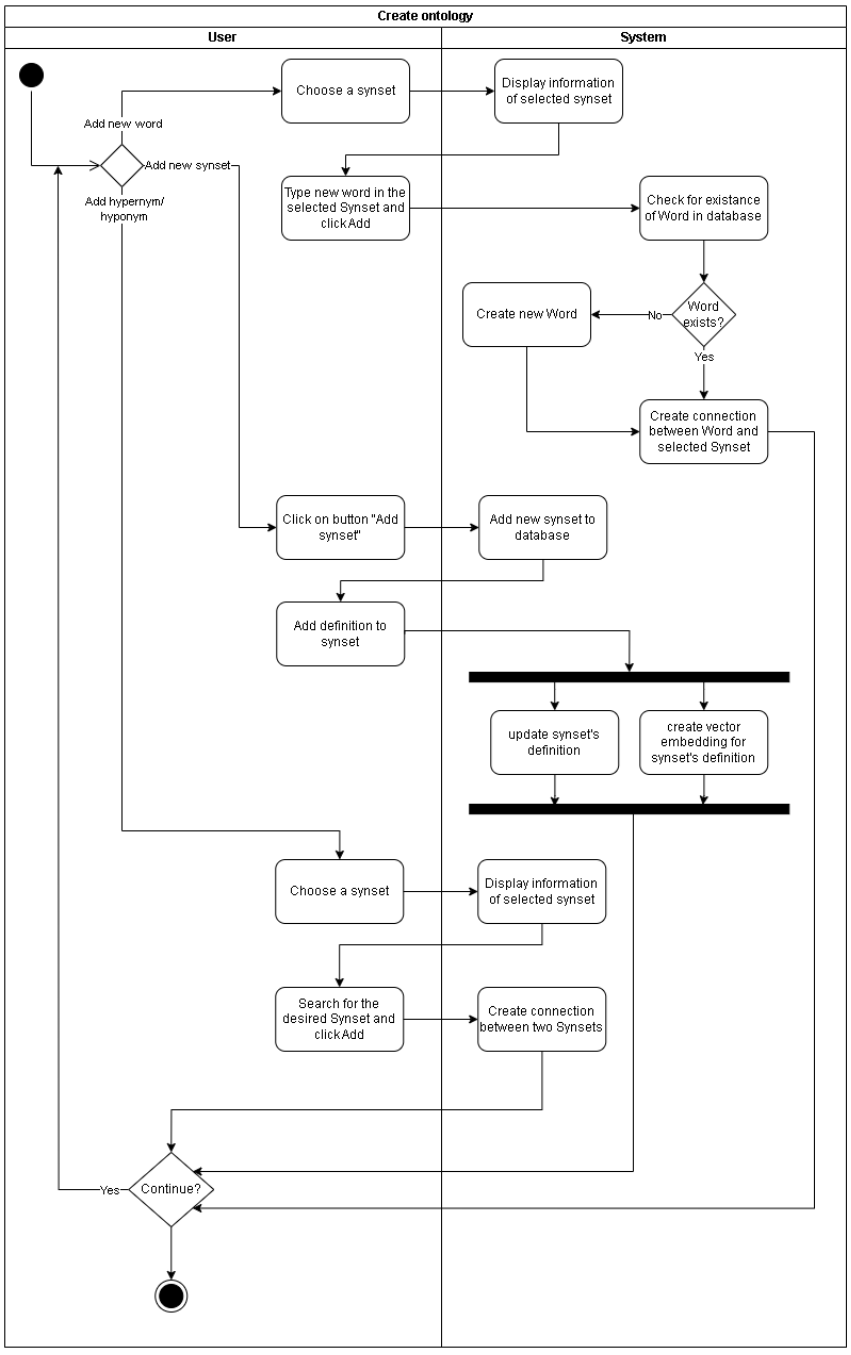

Figure 4.15: Activity diagram – Create ontology

While creating new ontology, user can perform actions such as adding new synsets, adding new words or creating connection between synsets. User can insert a synset by clicking button ”Add new synset”. Then user can give some definition to this synset and system will create vector embedding for this definition. User can add hyponym to this synset by searching for the

synset ID and click the ”Add” button, a connection will be formed between these two synsets. User can add set of synonyms (Word) to this synset by typing the synonym and click ”Add button”. System will check existance of the Word in the database. If the Word does not exist yet, it will be inserted to the selected Synset. User can repeat this process until the ontology completes.

4.4 Architecture design

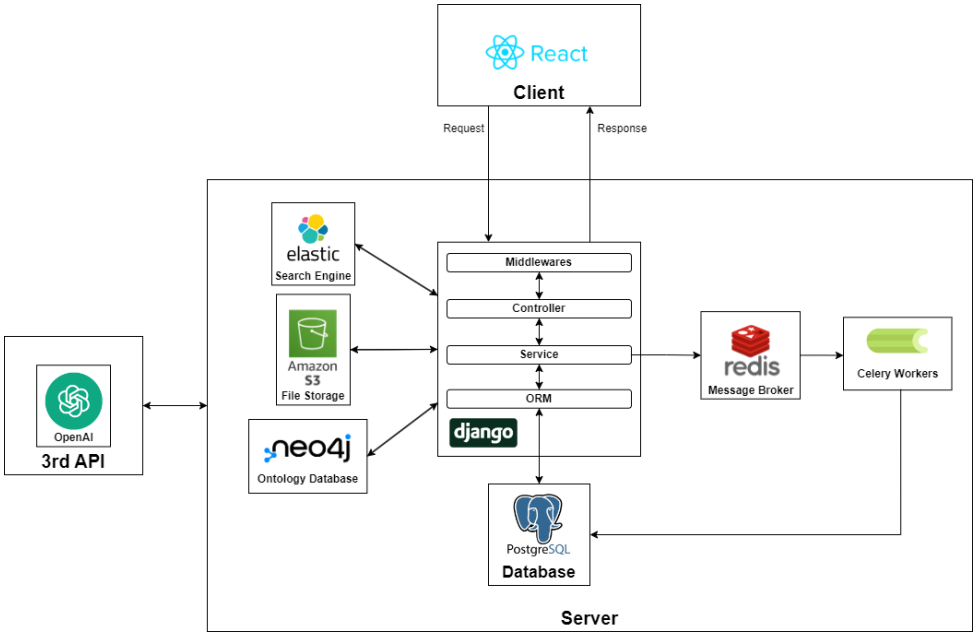

Figure 4.16: Overall System Architecture

Our software architecture follows a client-server pattern as illustrated in Figure 4.16, with the client built using ReactJS and the server utilizing Django along with additional components like Redis and Celery. This combination enables a robust, scalable, and asynchronous system.

a. Client: ReactJS

The client side is developed using ReactJS, a popular JavaScript library for building interactive user interfaces. React enables a responsive and dynamic user experience, making it well-suited for modern web applications.

b. Server: Django + Redis + Celery + PostgreSQL

Django:

- *Middleware Layer*: Incoming requests are processed through middleware components, handling tasks such as authentication, security checks, and request modification.
- *Controller Layer*: Django's URL dispatcher routes requests to the appropriate service layer to execute business logic.
- *Service Layer*: A separate service layer encapsulates the application-specific logic, interacting with models, databases, and external APIs. This promotes modularity and maintainability.
- *ORM (Object-Relational Mapping)*: Django's ORM, along with PostgreSQL, facilitates interaction with databases, abstracting away the underlying data storage details and providing a high-level interface for managing data.

Redis:

Redis serves as a cache backend, enhancing performance by storing frequently accessed data in-memory. It also acts as a message broker for Celery, facilitating the asynchronous execution of tasks.

Celery:

Celery handles asynchronous tasks, allowing the offloading of time-consuming operations to be processed in the background. It communicates with Redis as a message broker, picking up tasks initiated by the Django application. In our system, we use 2 concurrency workers to process task produced by Message Broker.

PostgreSQL:

PostgreSQL serves as the main relational database, providing a robust and scalable solution for storing and retrieving structured data. It integrates seamlessly with Django's ORM, ensuring efficient data management.

c. Communication: REST API

The communication between the ReactJS client and the Django server is established through a RESTful API. This enables seamless data exchange, supporting various client interactions with the server.

d. External APIs: AWS S3 and Elasticsearch

AWS S3:

AWS S3 is integrated to handle storage requirements, providing a scalable and secure solution for storing and retrieving document files.

Elasticsearch:

Elasticsearch is incorporated for advanced search capabilities and vector storage. It enhances data querying and indexing, supporting efficient and rapid search operations within the application.

OpenAI:

OpenAI is used for metadata extraction through prompting and to generate vector embeddings of document as well as user search query.

Neo4j:

Neo4j is the graph database used for storing and querying ontology. The graph structure inherent in the ontology renders Neo4j an appropriate choice for managing this task.

4.5 Database design

In this section, we present the database design – how we structure and organize data of the system. Initially, we design the conceptual model to illustrate entities, their behaviors and relationships between them. Following that, we detail the process of how we mapped this conceptual model into a physical model, incorporating tables as well as primary keys and foreign keys to signify the relationships between them.

4.5.1 Conceptual modeling

We design the conceptual model using Enhanced Entity Relationship Diagram (EERD)

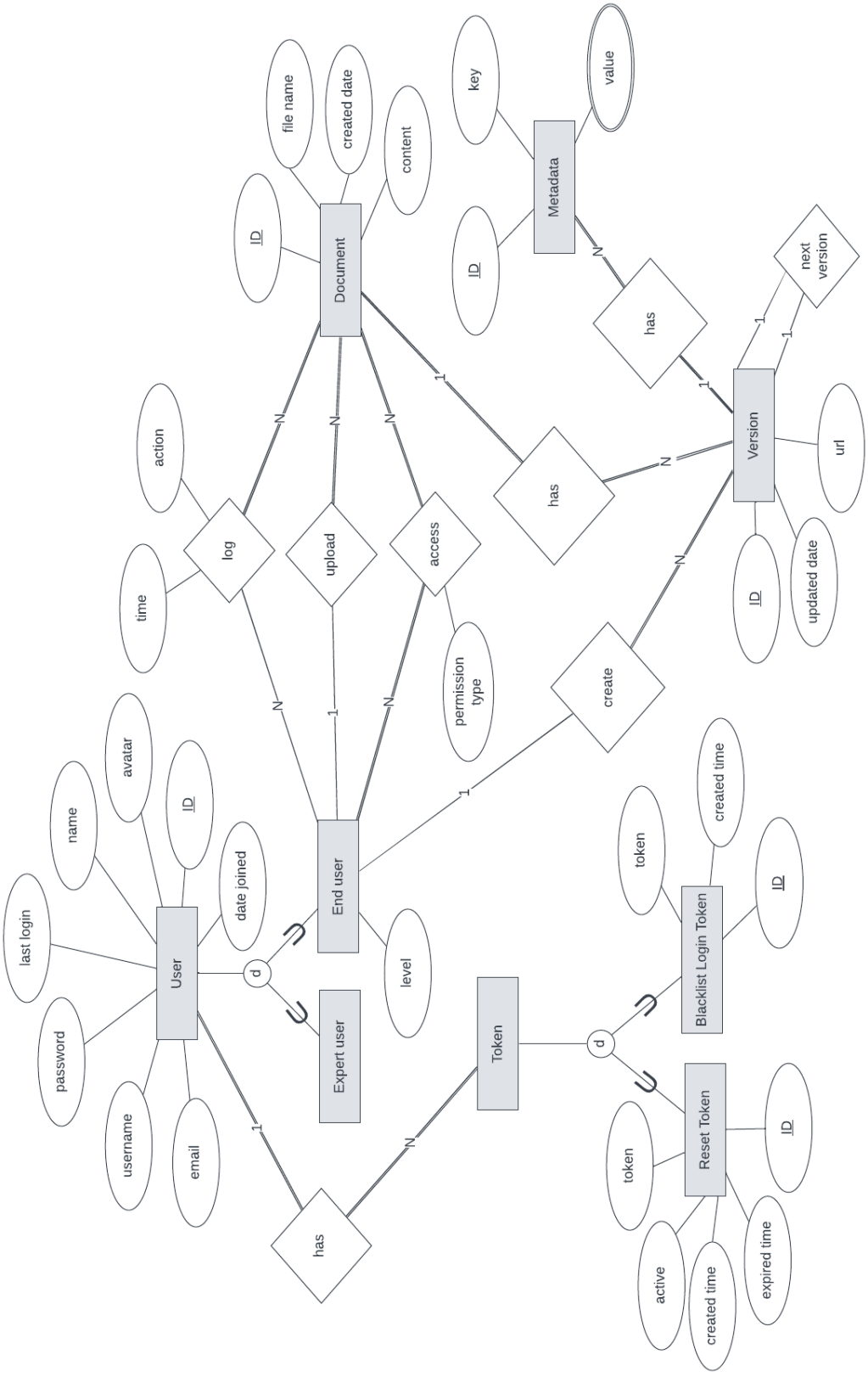

Figure 4.17: Conceptual model (EERD) of the database

Figure **4.17** illustrates the conceptual model of our database. The model mainly presents the relationship between User, Document, Ontology and their related entities. Next, we will analyze all entities and relationships in this model to see how we can map them into physical model.

Our database has the following entities:

- **User, Expert User, End User**: those entities are the main actors of our system, where Expert User and End User are the disjoint subclasses of User.

  Mapping into relational model: Considering that expert users do not have any additional attributes, and end users only have one additional attribute – level – which indicates their rank in the company, we decide to create only one table called "User" to contain information of all users. Beside background information of the user such as full name, email,… this table also contains an additional attribute called is_staff to indicate the role of user (Expert User or End User)

- **Token, ResetToken, BlacklistLoginToken**: those entities serve the purpose of authentication and authorization of our system, where ResetToken and BlacklistLoginToken are the disjoint subclasses of Token

  Mapping into relational model: In our system, authentication and authorization are proceeded frequently. Therefore, considering the nature of tokens, separating them into 2 separate tables would be an optimal choice so that the system can quickly retrieve the desired tokens. We create two tables which are ResetToken and BlacklistLoginToken, each has corresponding attributes suitable for their functions.

- **Document**: this entity stores data about documents that users upload to the system. It has basic information about a file which are file name, created date, content

  Mapping into relational model: Although being simple, this entity takes part in many relationships and activities of the system. First, we create table "Document" with the aboved attributes. We will get back and modify this table soon when it comes to relationships.

- **Version**: this entity stores data about different versions of a document. Its attributes are updated date and url (the url of the original file, which is stored on Amazon S3)

  Mapping into relational model: Like Document, we create table "Version" with those intial attributes and will come back to modify it later when relationships appear.

- **Metadata**: this entity stores data about metadata of all documents in the system. The entity has two attributes: key and value.

Mapping into relational model: Attribute value is a multi-valued attribute, indicating that a key in a metadata may have many values from different documents. Therefore, we decide to create two tables: "MetadataKey" and "MetadataValue". "MetadataKey" has one attribute called key. "MetadataValue" has two attributes: metadata_key_id (a foreign key referencing its key) and value (the value of the key it references)

Our database has the following relationships:

- **user has token**: this relationship contains information about which token belongs to which user. A user can have many tokens and a token only belongs to one user, so this is a one to many relationship.

  Mapping into relational model: Since this is a one to many relationship, we add a foreign key user_id (referencing "User") to both table "ResetToken" and "BlacklistLoginToken".

- **upload**: this relationship indicates the owner of the document (the user who uploaded the file). One user can upload many documents and one document can only be uploaded by one specific user. Therefore, this is a one to many relationship.

  Mapping into relational model: Traditionally, since "upload" is a one to many relationship, we can add a foreign key user_id (referencing "User") to table "Document" to express the owner of the document. And since "access" is a many to many relationship, we can create a new table which contains two foreign keys: one references "User" and one references "Document", to indicate access level of each user on each document. However, considering functional similarities between two relationships: both indicates some kind of permission of a user on a document. Therefore, we decide merge those relationships into one table by creating a new table called "DocumentPermission". This table has three attributes: user_id (foreign key, referencing "User"), document_id (foreign key, referencing "Document"), and permission (whose value may be either owner, view or edit). By doing this way, we reduce the complexity of the system, thereby reducing risks of data inconsistencies caused by inappropriate queries.

- **log**: this relationship logs the actions of a user on a document. Some actions such as viewing documents, downloading documents, modifying documents,… will be recorded by the system. One user can perform actions on many documents, and system can record actions of many users on one specific document. Therefore, this is a many to many relationship.

  Mapping into relational model: Since this is a many to many relationship, we create a new table called "ActivityLog" with four attributes: date, action, document_id (foreign key, referencing "Document") and user_id (foreign key, referencing "User").

- **document has version**: this relationship indicates the list of versions that a document may have. A document can have many versions, and one version must belong to a specific document. Therefore, this is a one to many relationship.

  Mapping into relational model: Since this is a one to many relationship, we add a foreign key document_id (referencing "Document") to table "Version".

- **next version**: this relationship indicates the versions order of a document. The versions must be sort in order from newest version to oldest version and user can restore the old version if they want to.

  Mapping into relational model: This data should be organized in a linked list. Therefore, we add a foreign key next_version_id (referencing "Version" itself) to table "Version".

- **version has metatada**: this relationship indicates the metadata that each version has. A new version will be created whenever user uploads a new file or modify document's metadata. A version may have many metadata value, and each metadata value must belong to only one version. Therefore, this is a one to many relationship.

  Mapping into relational model: Since this is a one to many relationship, we add a foreign key version_id (referencing "Version") to table "MetadataValue".

- **user creates version**: this relationship indicates the user who edits the document. The versions must be sort in order from newest version to oldest version and user can restore the old version if they want to. One user can create many versions of a document, and each version must be created by only one user. Therefore, this is a one to many relationship.

  Mapping into relational model: Since this is a one to many relationship, we add a foreign key user_id (referencing "User") to table "Version".

4.5.2 Relational modeling

With the mapping method that we have discussed in previous section, we design the physical modeling of our database system, illustrating in Figure **4.18**. In this physical design, the data type for each attribute of the table has also been specified. The tables are connected to each other using crow's foot notation in order to describe the cardinality of the relationships. Cardinality acts as a parameter for the relationship between entities. For one entity, there is a minimum and maximum number that helps define its relationship with another entity. With the present of cardinality in the modeling, we can detect fan trap and chasm trap right in the database desigining phase. While adding foreign key into table, we also consider whether that foreign key may cause any circular dependency in the design or not.

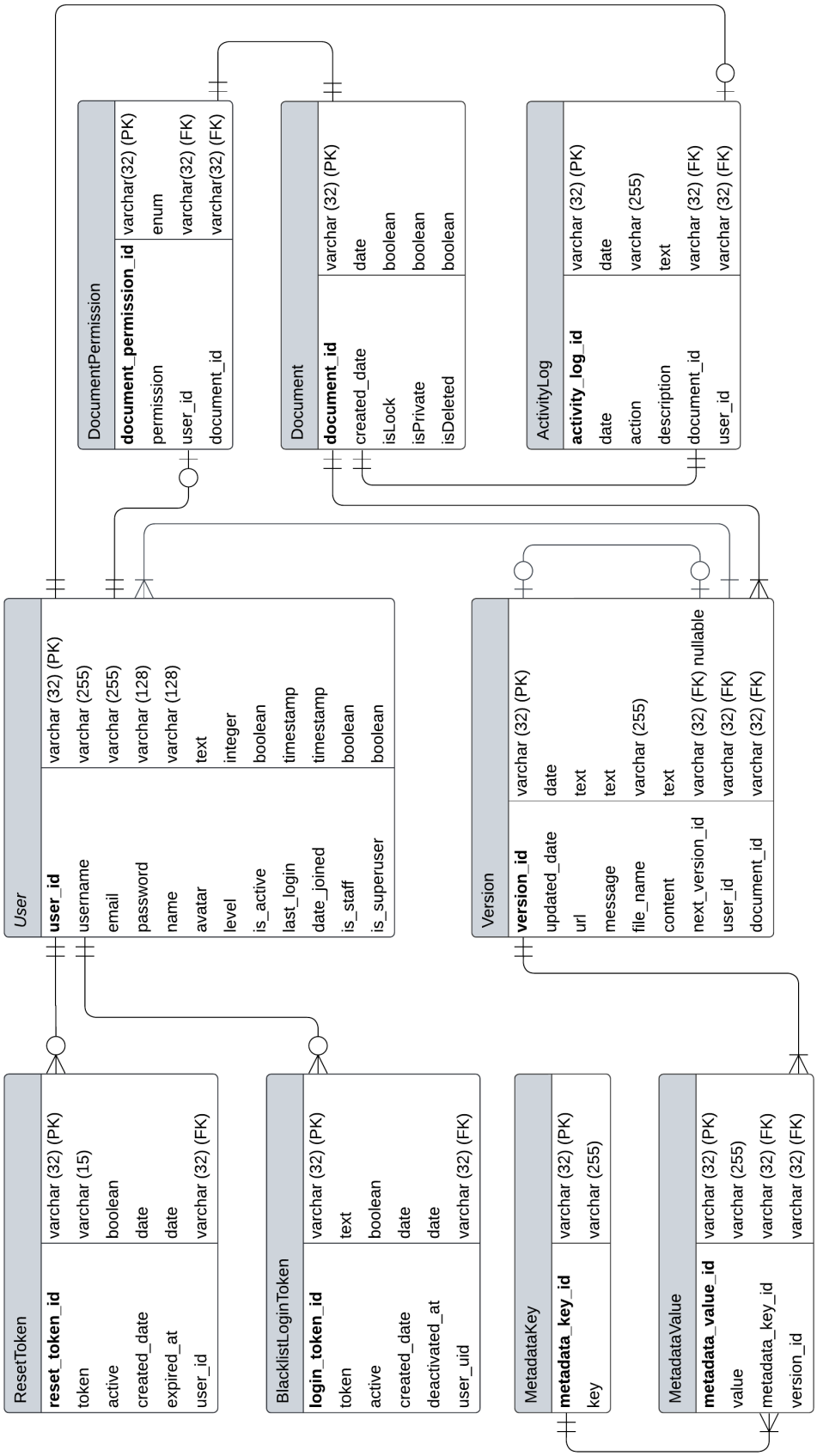

Figure 4.18: Relational model of the database

4.6 Metadata Extraction Implementation

4.6.1 Introduction

Initially, we explored a rule-based approach for this task, leveraging regular expressions (regex) to identify and extract specific information based on predefined patterns. However, the results were inefficient due to the variability in document formats and the dependence on the quality of Optical Character Recognition (OCR) outputs. Consequently, we pivoted towards a more advanced solution utilizing a Large Language Model (LLM), specifically GPT-3.5, to enhance the metadata extraction process.

4.6.2 Rule-Based Approach

a. Previous Implementation

Our initial approach to metadata extraction employed a rule-based method, utilizing regular expressions to define search criteria within semi-structured legal documents. This method relied on a set of predefined rules to identify patterns corresponding to specific metadata fields, as described in our previous report:

> ”Rule-based data extraction operates by relying on a predefined set of rules to extract specific information from a document. These rules serve as guidelines or instructions, dictating how the extraction process should identify and capture relevant data. After researching, we decided to choose regular expressions (regex) to define search criteria. Regular Expression (Regex) allows us to define patterns that match specific formats or structures within the document.”

b. Challenges and Limitations

Despite its theoretical simplicity, the rule-based approach presented several practical challenges:

- **High Dependency on Rule Design:** The effectiveness of the rule-based method was heavily contingent on the precision and comprehensiveness of the designed rules. Crafting these rules was a labor-intensive process, requiring extensive time and expertise, especially given the diverse formats of legal documents.
- **OCR Content Quality:** The accuracy of the extracted metadata was directly influenced by the quality of the OCR-processed content. Any noise or errors introduced during OCR processing could lead to misidentification or omission of metadata.

- **User Experience:** Users were required to manually select the type of legal document from numerous categories, resulting in a cumbersome and inefficient user experience.

4.6.3 Transition to LLM-Based Approach

a. Implementation with GPT-3.5

To address the shortcomings of the rule-based method, we adopted a machine learning-based approach using GPT-3.5. This transition was driven by the model's ability to understand and generate human-like text, thereby facilitating more accurate metadata extraction. By prompting the LLM to extract a predefined set of common metadata fields from legal documents, we leveraged its contextual understanding capabilities to process and interpret OCR-processed content.

b. Methodology

The implementation involved the following steps:

1. **OCR Processing:** Documents were first processed with OCR component to generate the content text.

2. **LLM Prompting:** The OCR-processed text was then input into GPT-3.5 with prompts designed to extract specific metadata fields. The prompt is presented in Fig 4.19, the prompt contains the predefined set of metadata to be extracted together with the type hinting and format to be returned, which is used to ensure consistency of the result.

```
model="gpt-3.5-turbo-0125",
messages=[
    {
    "role": "system",
    "content": [
        {
        "type": "text",
        "text": "Extract the possible document metadata from the legal document text provided
        (this is the first page of the document), return the result in Vietnamese with the
        JSON format \"metadata\":\"value\", correct any grammatical errors, this is the set of
        metadata with its type hinting in python, it must strictly follow this:\n\"Loại văn
        bản\": (type: str)\n\"Số\": (type: str)\n\"Ngày ban hành\": (type: str)\n\"Đơn vị ban
        hành\": (type: str)\n\"Căn cứ\": (type: List[str], top 3 relevant item)\n\"Nội dung\":
        (type: str)\n\"Lĩnh vực\": (type: str)"
        }
    ]
    },
    {
    "role": "user",
    "content": [
        {
        "type": "text",
        "text": content
        }
    ]
    },
    {
    "role": "assistant",
    "content": [
        {
        "type": "text",
        "text": "{\n    \"Loại văn bản\": \"\",\n    \"Số\": \"\",\n    \"Ngày ban hành\":
        \"\",\n    \"Đơn vị ban hành\": \"\",\n    \"Căn cứ\": [],\n    \"Nội dung\": \"\",
        \n    \"Lĩnh vực\": \"\"\n}"
        }
    ]
    }
],
temperature=0.1,
max_tokens=1024,
top_p=1,
frequency_penalty=0,
presence_penalty=0
```

Figure 4.19: Code for prompting

3. **Temperature Adjustment:** To minimize randomness and reduce the incidence of hallucinated outputs, the model was prompted with a low temperature setting, enhancing the consistency and reliability of the extracted metadata.

c. Evaluation and Results

The LLM-based approach demonstrated several advantages over the rule-based method:

- **Improved Accuracy:** GPT-3.5 was capable of correcting noise in the OCR output, yielding more accurate metadata extraction.
- **Contextual Understanding:** The model's ability to understand context allowed it to capture the correct metadata values even in varying document formats.

- **Efficiency:** This method significantly reduced the time and expertise required to design extraction rules, offering a more scalable solution.

However, the LLM approach was not without its challenges. Occasionally, the model produced metadata values that were hallucinated or irrelevant to the document content. To mitigate this issue, we set a low temperature during prompting to reduce the likelihood of such errors. Further analysis is presented in the Chapter 4: Evaluation.

4.7 Ontology building

For building the Vietnamese ontology for our system, we propose a novel approach by leveraging GPT, an advanced language models (LLMs), to translate synsets from the original Princeton Wordnet into Vietnamese. This methodology aims not only to bridge the lexical gap between English and Vietnamese but also to ensure semantic equivalence and cultural resonance in the translated ontology.

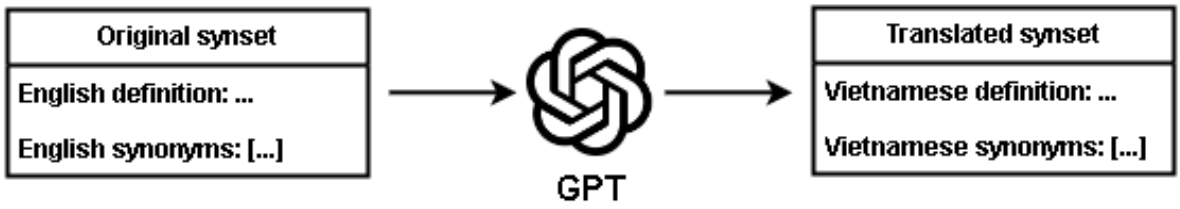

Figure 4.20: The process of translating ontology

Central to this method is the utilization of both the synset's list of synonyms and its definition as input for the translation process, as depict in **Figure 4.20**. By providing GPT with these two key components, we enable the model to grasp the nuanced meanings encapsulated within each synset, thereby facilitating a more accurate translation. In traditional translational method, if a word to be translate has many meanings but the definition of that word is not provided, then usually it will be translated to the most popular meaning, which may lead to inaccuracy if the desired term is actually the less popular meaning. For example, if we translate the word *"bank"* without specific context, usually we will get *"ngân hàng"* as the result, which is inaccurate if the intended meaning of "bank" we refer to is *"bờ sông"*. To get the result as *"bờ sông"*, we need to provide some definition to LLM, such as *"the side of a river, canal, etc. and the land near it"*. Therefore, providing both the synset's list of synonyms and its definition to LLM is neccessary to ensure that all translated synonyms are semantically equivalent to the synset's definition.

Unlike traditional translation methods that may focus solely on word-by-word substitution, our approach recognizes the inherent complexity of language and the need for semantic fidelity.

Thus, we instruct the LLM to prioritize conveying the essence and meaning of the synset rather than adhering strictly to a one-to-one translation of individual words. This flexibility allows for the adaptation of terms and expressions to better resonate with Vietnamese speakers while maintaining semantic coherence. Therefore, the key innovation of our approach lies in the acknowledgment that the translated synonyms array does not need to mirror the exact composition of the original English synonyms. Instead, we emphasize the importance of semantic equivalence, encouraging the LLM to generate synonyms that convey similar meanings and connotations in Vietnamese, even if they are not direct translations. Consider this example to illustrate the point:

Original English Synset:

- *Synonyms*: auto, automobile, car, machine, motorcar
- *Definition*: A wheeled motor vehicle used for transporting passengers, which also carries its own engine or motor.

Direct Translation:

- *Synonyms*: tự động, ô tô, xe hơi, máy móc, xe ô tô
- *Definition*: Một phương tiện có bánh xe được sử dụng để vận chuyển hành khách, đồng thời cũng mang theo động cơ riêng hoặc motor.

Adapted Translation:

- *Synonyms*: xe hơi, ô tô, xe con
- *Definition*: Một phương tiện di chuyển có bánh xe được sử dụng để vận chuyển hành khách, thường trang bị động cơ riêng hoặc motor.

In the direct translation, the individual English synonyms are replaced with their corresponding Vietnamese counterparts, resulting in a rigid translation. While this conveys the basic meaning of the original synset, it may lack the nuanced semantic equivalence required for an effective Wordnet entry in Vietnamese. Moreover, some translated terms fail to capture the real meaning of the synset (auto -> tự động, machine -> máy móc)

In the adapted translation, the emphasis is on conveying the essence and function of the original synset in a way that resonates with Vietnamese speakers. The synonyms have been adjusted to capture the concept of ”car”, reflecting the cultural context and usage of language in Vietnamese. Additionally, the definition has been refined to accommodate variations in terminology and to ensure clarity and relevance within the target linguistic community.

Moreover, cultural nuances play a pivotal role in ensuring the relevance and comprehen-

sibility of the translated Wordnet within the Vietnamese context. Therefore, our methodology incorporates considerations of cultural sensitivities and linguistic idiosyncrasies, guiding the LLM to produce translations that are not only linguistically accurate but also culturally appropriate.

By adopting this methodological framework, we aim at building a Vietnamese ontology that effectively gather as many words in the Vietnamese vocabulary as possible, correctly assign them to appropriate synsets that accurately represent their meanings, thereby can be used in our document management system. The resultant ontology built by this method has a total of 98303 synsets, 133051 distinct words and 197231 senses.

4.8 Query the ontology

The primary objective of this ontology is to find relevant keywords which are semantically equivalent to the original terms that appear in user's search query. In this section, we discuss the process of querying the ontology to identify those keywords, using Neo4j's CYPHER language in conjunction with techniques including tokenization and sense disambiguation.

4.8.1 First attempt - using substring matching

To identify all Words in the ontology that appear in the user's search query, we utilize the CYPHER query in code snippet **4.1**. In this query, `(target:Word)` represents nodes `target` of the type `Word`, and `$user_query` is the variable containing the user's search input. The `CONTAINS` operator is used for substring matching, enabling the query to return all Word nodes where `target.label` (the word itself) is found within the `$user_query`.

```
MATCH (target:Word) WHERE $user_query CONTAINS target.label
RETURN target
```

Listing 4.1: CYPHER using CONTAINS operator

This approach ensures that all relevant Word nodes are accurately retrieved based on the user's search criteria. However, since Neo4j does not support indexing for this type of retrieval and due to the large scale of the ontology, the query execution time is notably high. Consequently, we propose an alternative approach, which is discussed in the subsequent section.

4.8.2 Tokenization and precise matching

Neo4j features `TEXT INDEX` that supports queries utilizing the "=" operator for checking string equality. This capability allows for precise matching of entire strings, as opposed to the

CONTAINS operator, which is used for substring matching. To implement this method, first we need to tokenize the user's search input.

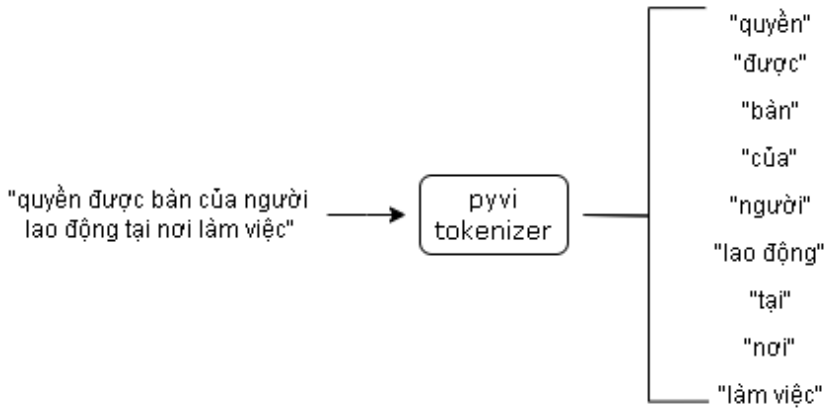

Figure 4.21: Tokenizing the user's search input

In this method, we use pyvi [26], a Python Vietnamese Toolkit which facilitates various linguistic tasks including tokenization, part-of-speech (POS) tagging, accent removal, and accent addition. Given a user's search input such as *"quyền được bàn của người lao động tại nơi làm việc"*, the input undergoes tokenization, resulting in a structured array of semantically meaningful units. This process is exemplified in figure 4.21, illustrating the breakdown of the input string into its constituent components.

However, a notable limitation of this tokenization approach is its tendency to fragment the string into the smallest meaningful units, potentially overlooking opportunities to combine words into coherent phrases. For instance, in the illustration provided in Figure **4.21**, the words *"người"* and *"lao động"* could be effectively grouped together as *"người lao động"*, enhancing the phrase's semantic clarity. Similarly, *"nơi"* and *"làm việc"* could be concatenated to form *"nơi làm việc"* - an even more cohesive phrase. To address this challenge, we integrate a 2-gram technique, concatenating adjacent tokens to create a new array comprising both the original tokens and the newly combined adjacent tokens. This ensures that coherent phrases, such as *"người lao động"* and *"nơi làm việc"*, are preserved and not overlooked after tokenization. The array of words resulting from this process is *['quyền', 'được', 'bàn', 'của', 'người', 'lao động', 'tại', 'nơi', 'làm việc', 'quyền được', 'được bàn', 'bàn của', 'của người', 'người lao động', 'lao động tại', 'tại nơi', 'nơi làm việc']*.

```
UNWIND $word_list AS term
CALL{
    WITH term
    MATCH (target:Word) WHERE target.label = term
    WITH target
```

```
    MATCH (syn:Synset)<-[r:BELONGS_TO]-(target)
}
```

Listing 4.2: CYPHER - retrieve all Words that match the terms in word_list

The words array is given as input parameter to the CYPHER snippet code **4.2**. This retrieval query uses precision matching operator `MATCH (target:Word) WHERE target.label = term` instead of `CONTAINS` operator in the first attempt. This query utilizes Neo4j's `TEXT INDEX`, thereby enhances query execution time significantly.

4.8.3 Sense disambiguation

As in code snippet **4.2**, the initial step involves retrieving all Word nodes that match the term specified in the user's query. Subsequently, a supplementary query is conducted to get the Synset node(s) to which these Word nodes belong to. It is crucial to acknowledge that a single Word may have multiple meanings, thereby it may belong to several Synsets. For instance, the word *"bàn"* embodies two distinct meanings: as a noun, signifying *"table"*, and as a verb, connoting *"to discuss"*. Consequently, it belongs to two distinct synsets, one delineating *"a piece of furniture having a smooth flat top that is usually supported by many vertical legs"*, while the other characterizes *"an act of talking about something in detail, showing the different ideas and opinions about it"*. In the context of our user's search input, *"quyền được bàn của người lao động tại nơi làm việc"*, the usage of *"bàn"* distinctly has the meaning of *"to discuss"* from a human perspective. However, for the system to accurately discern the intended meaning of a word, the integration of a sense disambiguation mechanism is neccessary.

There are many approaches for sense disambiguation, including decision trees, neural network, dictionary-based,... In our ontology, given that each Synset has a *"Definition"* attribute used to briefly describe the concept of the synset, vector embedding - a NLP method used for text analysis - is a suitable approach for the purpose of sense disambiguation. Vector embedding is the method that encodes the meaning of words into a real-valued vector. Words that are closer in the vector space are expected to be similar in meaning. There are several methods to quantify the similarity between vectors, including Cosine Similarity, Euclidean Distance, Manhattan Distance,... In our approach, we leverage OpenAi's text-embedding model to embed the *"Definition"* attribute of each Synset into vectors (figure **4.22**). When CYPHER confronts a Word that belongs to many Synset (which means that Word has many meanings), our approach selects the Synset with highest cosine similarity between its embedding and the search input's embedding of the user, ensuring that the keywords returned are semantically equivalent to user's input.

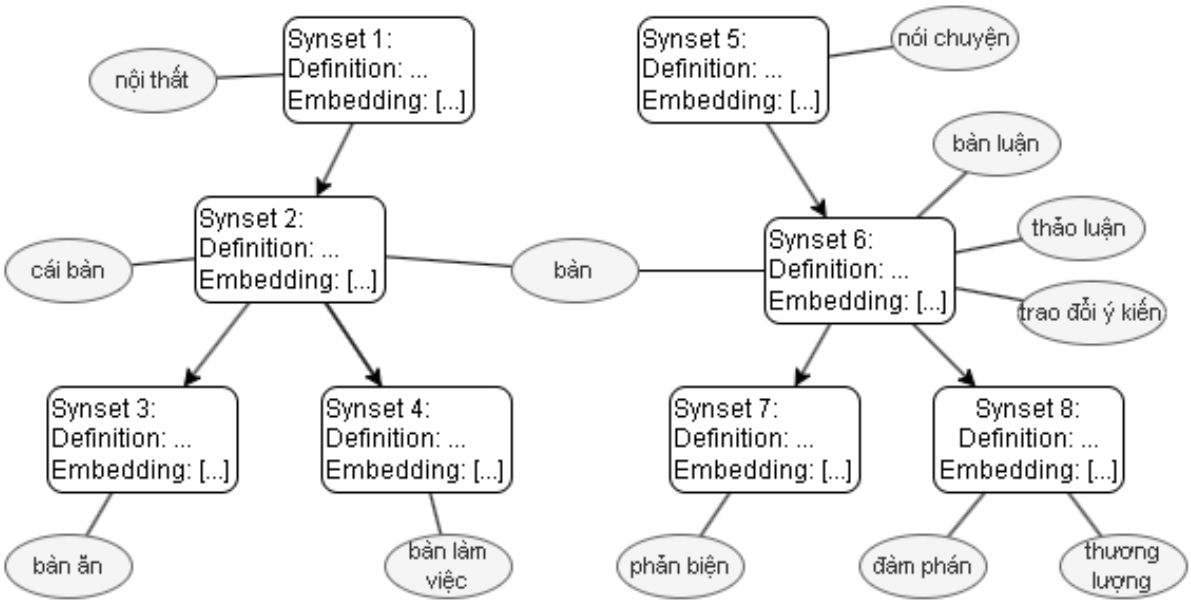

Figure 4.22: Ontology with vector embedding

4.8.4 Retrieve synonyms, broader, narrower terms

When the synset to which each term from the user's search input belongs to is determined, the system proceeds to query the ontology, extracting synonymous terms linked to that identified synset (referenced as line 3 in code snippet **4.3**). Furthermore, the hypernyms (broader concept) and hyponyms (narrower concept) of that synset are also retrieved (referenced as lines 4 to 13 in code snippet **4.3**).

```
1  WITH syn
2  CALL {
3      MATCH (synonym:Word)-[:BELONGS_TO]->(synset)
4      OPTIONAL MATCH (hypernym:Synset)-[:HAS_HYPONYM]->(syn)
5      WITH hypernym
6      CALL{
7          MATCH (broader:Word)-[:BELONGS_TO]->(hypernym)
8      }
9      OPTIONAL MATCH (hyponym:Synset)<-[:HAS_HYPONYM]-(syn)
10     WITH hyponym
11     CALL{
12         MATCH (narrower:Word)-[:BELONGS_TO]->(hyponym)
13     }
14     RETURN syn, synonym, hypernym, broader, hyponym, narrower
15 }
```

Listing 4.3: CYPHER - retrieve synonyms, broader, narrower terms of words from word_list

The retrieved synonymous keywords inherently share semantic equivalence with the terms

provided by the user, so they are expanded into original query as interchangeable keywords, thereby augment user's query. While broader and narrower terms may not possess identical semantic equivalence to synonyms, they still have closely related meanings to the user's terms. Therefore, these terms are suggested to user as optional expansions for further query refinement. An example outcome of this process are illustrated in Figure **4.23**, depicting the user's original query, the expanded query incorporating synonyms, and the broader/narrower terms suggested for optional expansion.

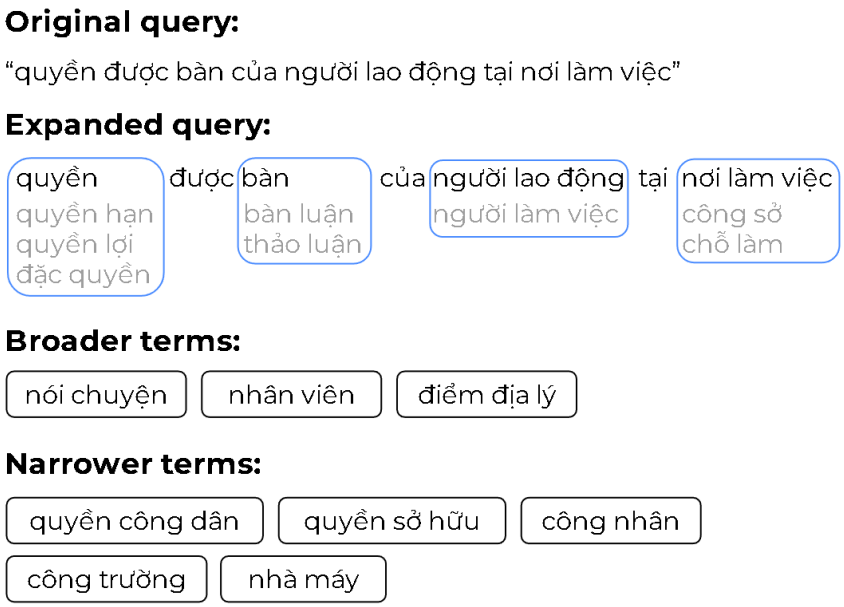

Figure 4.23: Query after refined by ontology

4.9 Search Component Implementation

4.9.1 Overview

In our document management system, we have implemented three distinct search methods to enhance the retrieval of relevant information: Full-text Search, Query Expansion with Domain Ontology, and Semantic Search.

Each method provides unique capabilities that address different aspects of search efficiency and accuracy, ensuring comprehensive and precise results for users. The need for these diverse search methods comes from the complex nature of Vietnamese languages, which often contains nuanced language, specialized terminology, and require contextual understanding.

We believe that with these search methods, we can address various scenarios users may encounter and efficiently retrieve relevant information from such documents required advanced

search mechanisms that go beyond simple keyword matching. Different search methods address various challenges:

- **Full-text Search**: Provides a straightforward approach to find documents containing specific terms or phrases.
- **Query Expansion with Domain Ontology**: Enhances full-text search capabilities by expanding user queries with related terms and concepts using a Domain Ontology, ensuring a broader and more accurate search result.
- **Semantic Search**: Understands the context and meaning behind search queries, allowing for more precise and contextually relevant documents.

The three search methods complement each other by addressing different aspects of search requirements:

- **Full-text Search** serves as the foundational search method, offering quick and direct results for specific term queries. However, it may miss documents that use different terminology or phrasing.
- **Query Expansion with Domain Ontology** complements Full-text Search by broadening the search scope. It ensures that related terms and synonyms are considered, capturing documents that may be relevant but use specialized terminology.
- **Semantic Search** adds a layer of contextual understanding, interpreting the meaning behind queries and providing results that are contextually relevant. It bridges the gap when exact terms are not known or when the query involves complex legal concepts.

4.9.2 Full-text search

Full-text Search is a search method that allows users to search for specific terms or phrases within the text of documents. It is based on matching the exact terms entered by the user against the content of the documents. This method is particularly useful for locating documents that contain specific keywords or phrases quickly and efficiently.

Full-text Search operates based on an inverted index. An inverted index is a data structure used to map terms (words or phrases) to their locations within a set of documents. This index allows for rapid searching because it organizes data in a way that makes it easy to find all occurrences of a term across a large corpus of text.

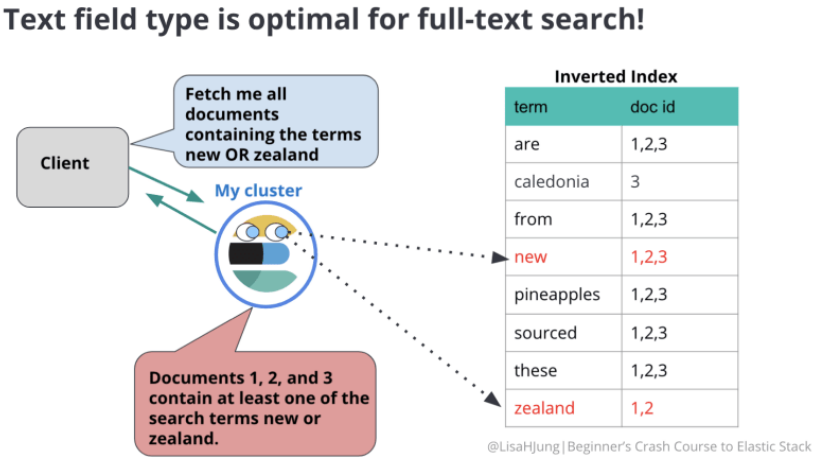

Figure 4.24: How full-text search works with inverted index

Figure 4.24 explain how full-text search in Elasticsearch works with the below process:

- **Term Extraction**: During the indexing process, each document is analyzed, and its text content is broken down into individual terms or tokens.
- **Inverted Index Creation**: For each term, an index entry is created that lists all the documents and positions within those documents where the term appears.
- **Efficient Lookup**: When a search query is submitted, the inverted index allows the search engine to quickly locate and retrieve documents that contain the query terms.

Our system conducts full-text search across all document text fields, including content and metadata. Utilizing the BM25 scoring algorithm, documents are ranked based on relevance, considering factors like term frequency and document length.

4.9.3 Query expansion with Ontology

Query expansion with ontology enhances search capabilities by expanding user queries with related terms and concepts based on a predefined domain ontology—a structured framework of knowledge. By expanding the initial query with related terms and synonyms, Query expansion aims to increase the likelihood of retrieving comprehensive and relevant search results. This part will not explain how to build a domain ontology but rather how the predefined domain ontology is utilized to expand the user query.

Method: The process of Query Expansion in our system start with expanding the initial query with related terms and synonyms then executing the query just as the Full-text Search method, below is the detail process:

1. **Identifying the Original Term:** The system begins by extracting the original term or terms from the user query. This forms the basis of the search query.

2. **Finding Related Terms:** Next, the system searches its database or knowledge base to identify related terms and synonyms associated with the original term. This step may involve various techniques, such as leveraging ontologies, semantic networks, or statistical methods, to determine the semantic similarity between terms.

3. **Combining Original and Related Terms:** Once the related terms are identified, the system combines them with the original query using Boolean operators, typically the OR operator. This creates a new expanded query that includes both the original term and its related terms. Ex: "term1 term2" will become "(term1 OR related_term1) term2", a more detail version is in Fig 4.25

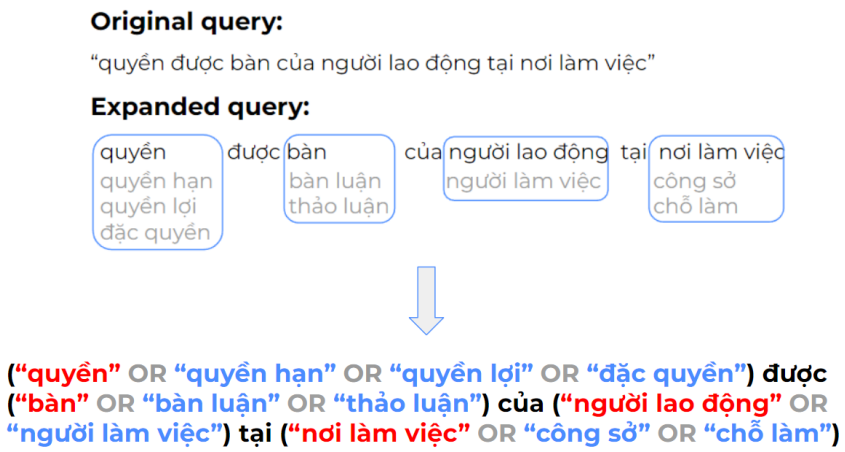

Figure 4.25: How Query Expansion works

4. **Executing the Expanded Query:** The system then executes the expanded query against the document repository, using the same method as full-text search, retrieving documents that match any of the terms included in the expanded query.

In our query expansion method, we employ the same BM25 scoring mechanism as used in full-text search. This ensures consistency in scoring across both the original query and the expanded terms.

4.9.4 Semantic search

Semantic Search is an advanced search technique that goes beyond traditional keyword matching to understand the context and meaning behind user queries. At its core, semantic

search relies on dense vector representations of text, which capture the semantic similarities between words, phrases, and documents.

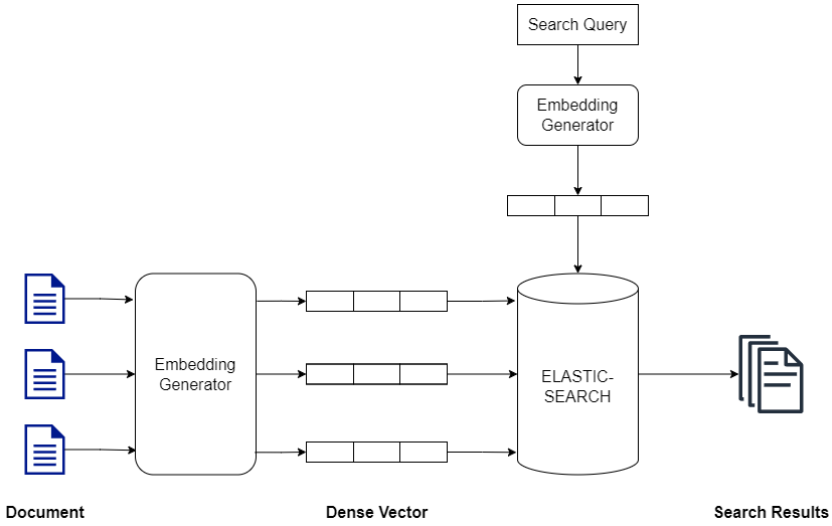

Figure 4.26: How semantic search works

According to Figure 4.26, the implementation of semantic search involves several key steps:

1. **Producing Vector Embeddings**: When a document is submitted for indexing, its content is processed for generating embeddings by the Embedding generator. The document is chunked into smaller segments or paragraphs, ensuring that each chunk represents a coherent unit of text. This chunking process allows for more efficient processing of large documents and helps capture the semantic nuances present within the text. Once the document is chunked, vector embeddings are generated for each chunk using the pre-trained language model. These embeddings capture the semantic meaning of the text within each chunk, encoding information about the words, phrases, and their relationships.

2. **Storing Vector Embedding Chunks on Elasticsearch**: After generating vector embeddings for each chunk, they are stored in an Elasticsearch index. Each chunk is associated with its corresponding vector embedding, allowing for quick and efficient retrieval during search queries.

3. **Generating Embedding for User Query**: When a user submits a search query, the query text is first converted into a dense vector embedding using the same embedding model.

4. **Search and Ranking Based on Similarity of Embedding**: The vector embeddings of the user query are then compared to the embeddings of the document chunks stored in Elasticsearch. This is done using **Cosine Similarity**, a measure of similarity between two vectors that takes into account the angle between them. Documents with embeddings

most similar to the query embedding are ranked higher in the search results.

5. **Returning the Top Results**: Finally, the top-ranked documents based on their semantic similarity to the user query are returned as search results.

To support the implementation of semantic search in our document management system, specific settings and configurations are required within Elasticsearch. The provided configuration defines the mapping for the Elasticsearch index, specifying how document chunks and their corresponding vector embeddings should be indexed and queried:

```
{
  "mappings": {
    "properties": {
      "chunks": {
        "type": "nested",
        "properties": {
          "chunk_content": {
            "type": "text"
          },
          "chunk_embedding": {
            "type": "dense_vector",
            "dims": 768,
            "index": true,
            "similarity": "cosine"
          }
        }
      },
      "content": {
        "type": "text",
      },
    }
  }
}
```

- **Chunks:** This property represents the chunks of text into which large documents are divided. It is defined as a nested type to maintain the relationship between the chunk content and its corresponding vector embedding.
  - **Chunk Content:** This sub-property stores the actual text content of each chunk. It is defined as a text type to support full-text search within individual chunks.
  - **Chunk Embedding:** This sub-property stores the dense vector embedding of each chunk. It is defined as a dense_vector type with a specified dimensionality (*dims*: 768), indicating the length of the vector. Additionally, it is configured to be indexed

(*index*: true) to enable efficient similarity search, and similarity is calculated using the cosine similarity metric (*similarity*: cosine).

- **Content:** This property represents the full-text content of the document. It is defined as a text type to support full-text search across the entire document.

4.10 User interface

Document feed

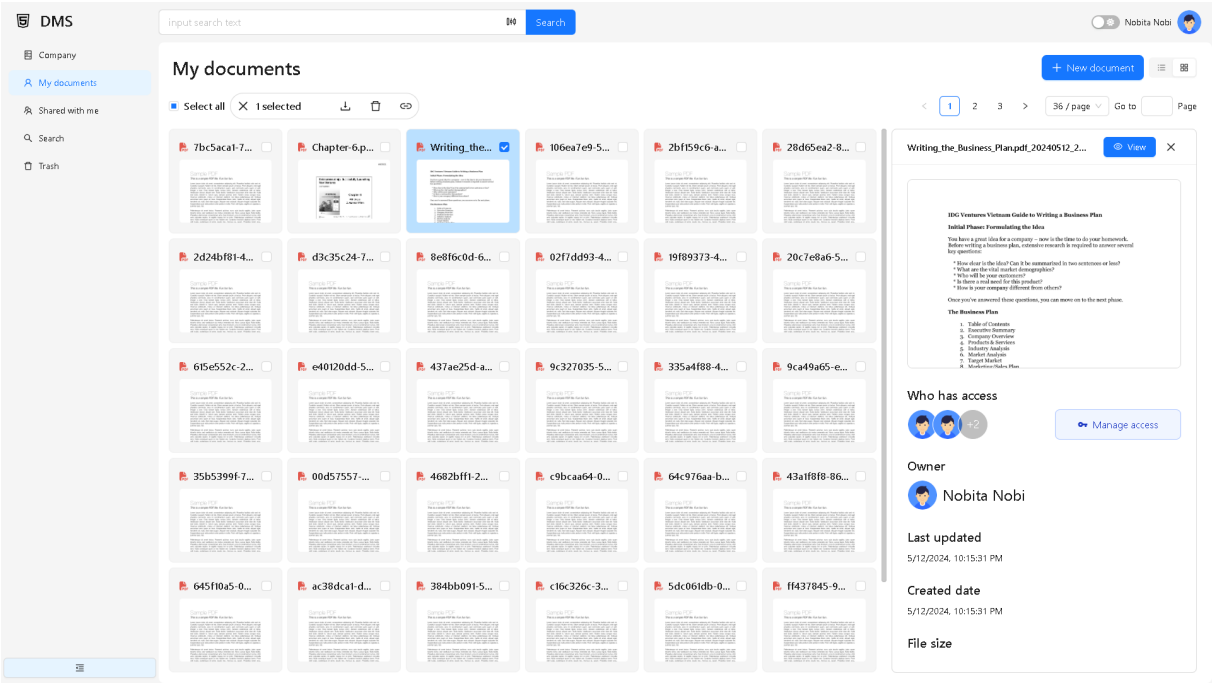

Figure 4.27: User interface - Document feed (grid view)

The document feed of the system (figure **4.27** is designed to enhance user interaction and document accessibility. The grid view layout presents documents as individual cards, providing a succinct visual summary of each file. This design choice not only facilitates quick identification but also simplifies the navigation process, allowing users to efficiently manage their document repository.

Key features of the interface include:

- **Grid View Presentation**: Documents are organized in a grid format, offering a structured and visually appealing overview.
- **Document Cards**: Each document is encapsulated within a card, displaying a preview of the content for easy recognition.

- **Interactive Elements**: Users can perform multiple document management tasks, such as selection, deletion, and organization, with ease.
- **Search Bar**: A dedicated search field on the nav bar enables swift filtering and retrieval of documents from any page in the system.
- **Detailed Information Panel**: Upon selecting a document, a side panel reveals important information about that document, including created date, updated date, people who can access,... aiding in the effective management of documents.

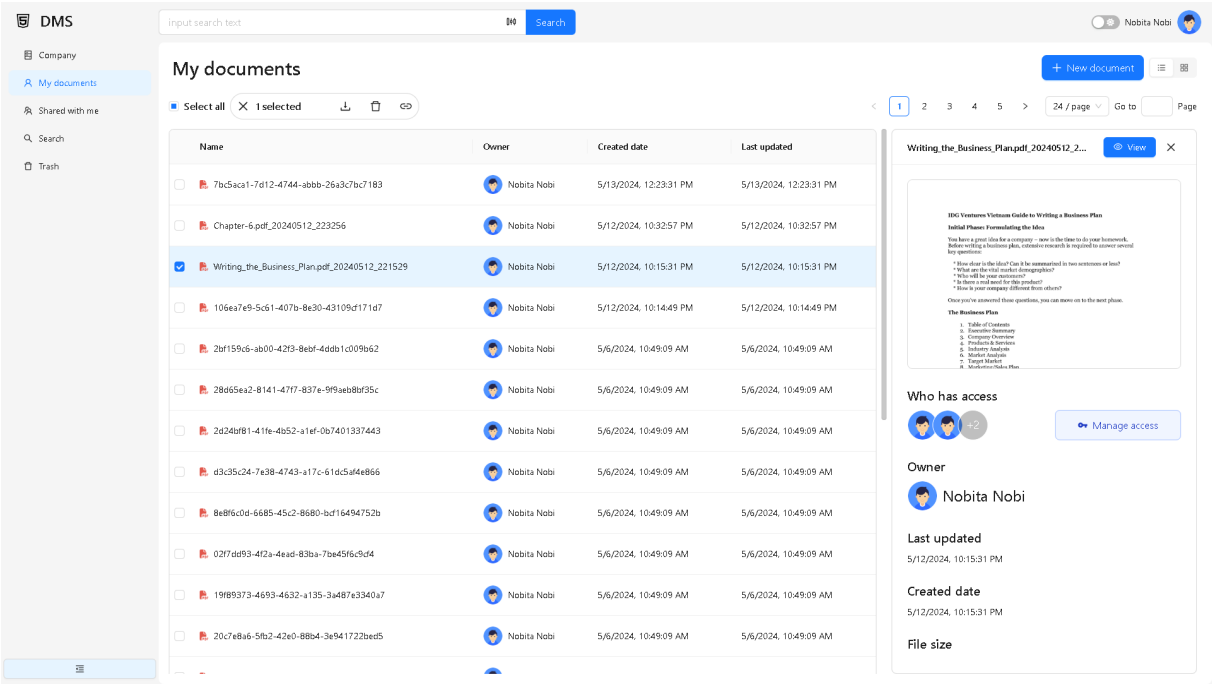

Figure 4.28: User interface - Document feed (list view)

Figure **4.28** represents the list view of the document feed. The main panel displays a list of all the user's documents, including the document name, creation date, last updated date, and file size. Users can sort the documents by any of these criteria - such as the date the document was created or last updated - by clicking on the column headers.

Upload document

To upload a document, users get through 3 steps:

1. Upload draft file
2. OCR - extract metadata
3. Edit metadata - save to cloud

Figure 4.29 is the modal for uploading documents. On top of this modal is a stepper that shows the steps that users have to get through, as well as the current step. Figure 4.29 shows step 1 of the uploading process. In step 1, users upload the file from their device, which will be rendered if successfully uploaded. After that, users can start the OCR process by clicking on the "Start OCR" button.

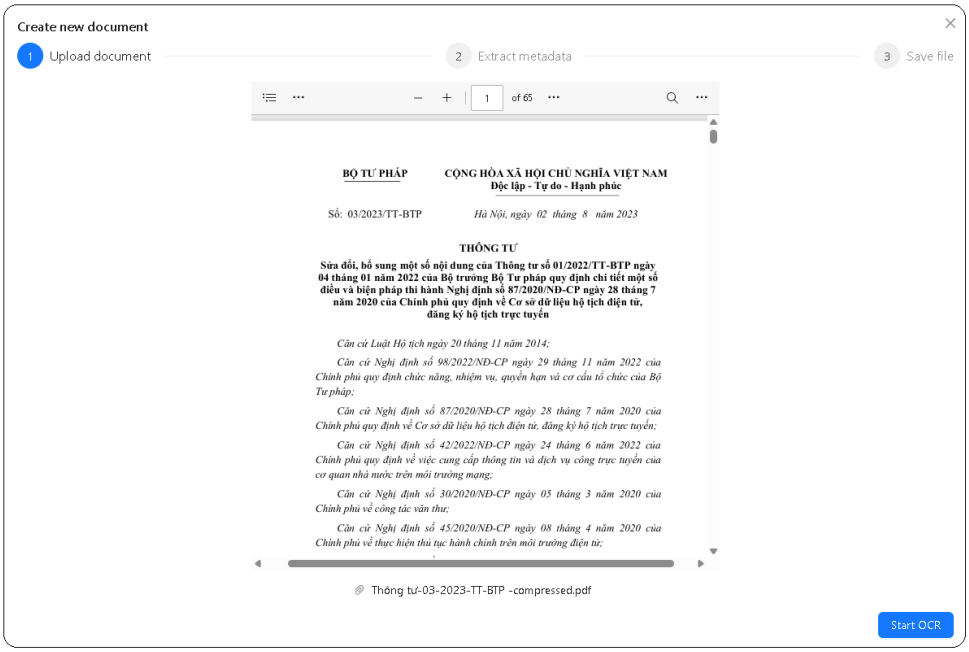

Figure 4.29: User interface - Upload document (step 1)

Figure 4.30 shows step 2 of the uploading process. On this step, system implements the OCR process to extract content from the document. The OCR process may take some time, depending on the size and complexity of the document.

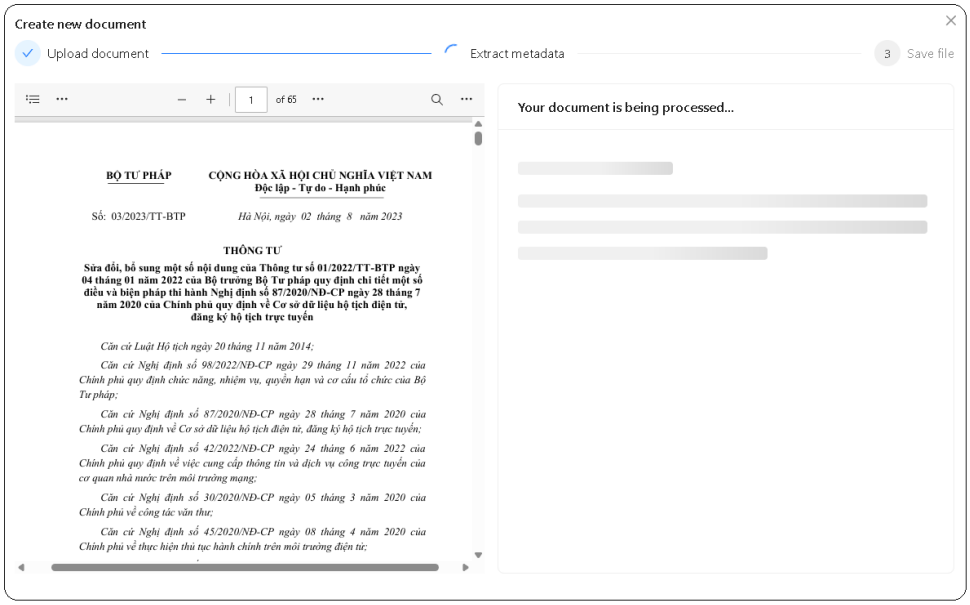

Figure 4.30: User interface - Upload document (step 2)

Figure 4.31 shows the final step of the uploading process, when the metadata of the document have been extracted completedly. In this step, the document is shown on the left, with the result metadata on the right. The metadata fields are editable, and users can also add new metadata by entering a new pair of key - value. Users can compare the OCR result with the original document and make corrections if needed. After making all the changes, users can click on the "Save Document" button to store the new document and its metadata to the database and to cloud service.

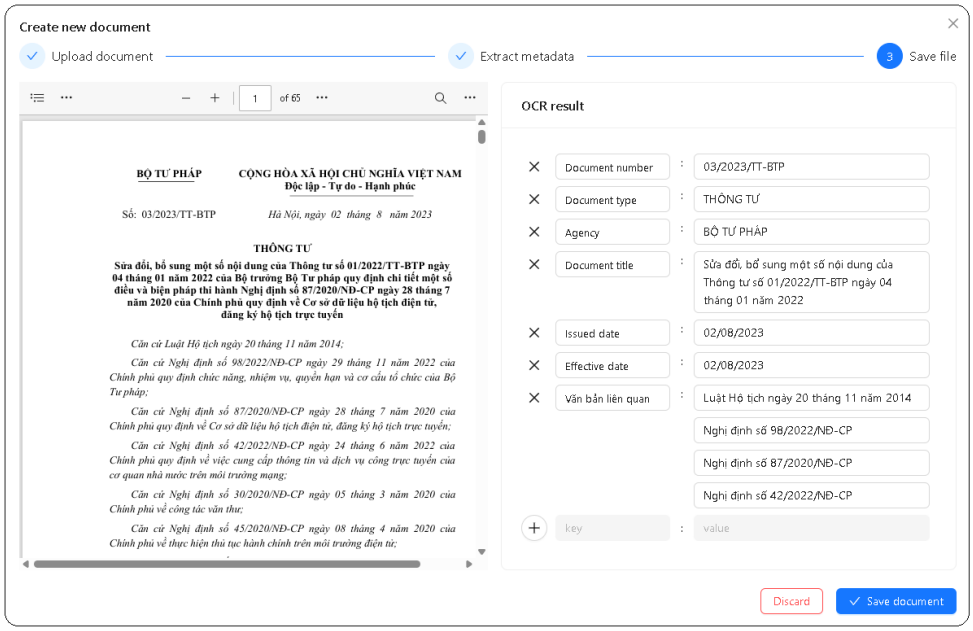

Figure 4.31: User interface - Upload document (step 3)

The uploading process may take a while, particularly considering potential high executable time in OCR procedures. To address this concern, we underscores the strategic utilization of a modal interface. Through this interface, users have the capability to concurrently engage in tasks across multiple documents while their uploading materials still undergo OCR processing in the background. This approach optimizes user productivity and workflow continuity, thereby enhancing the overall user experience within the document management system.

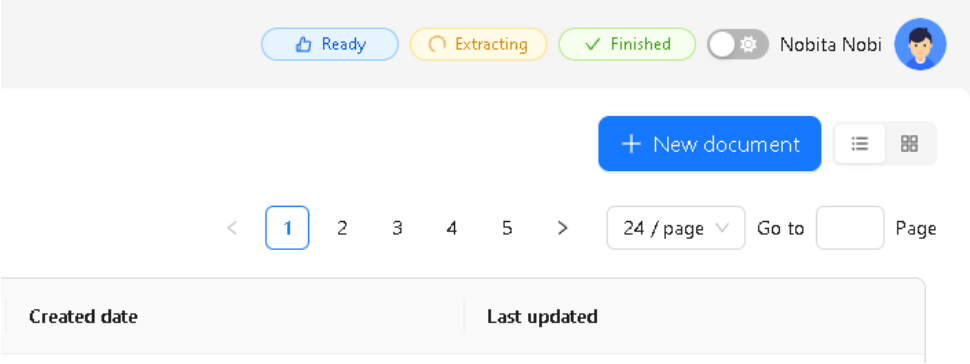

Figure 4.32: User interface - Upload multiple files

The system also allow users to upload multiple files at once. Whenever users upload a new file, a colorful tag appears on the nav bar at the top right corner of the interface (figure **4.32**. This tag has three states corresponding to three steps in the uploading process. State "Ready" shows that the file is ready for OCR. State "Extracting" indicates that the file is undergoing OCR process. State "Finish" notifies that metadata has been extracted successfully and users can make changes to the metadata and finally save them to the database. With this function, users can upload multiple files separately without interfering other processes.

View document details

Figure **4.33** is the interface that display details about a document. On the left side, there is a preview of the document content, allowing users to read and review the document directly within the interface. The right side is a file information section that shows the file name, size, owner, and creation date, with an option to rename the file. Below this section is the metadata section that provides detailed information about the document like document number, type, agency, effective date, and other attributes, with an option to update these details. At the bottom are buttons for various actions, such as manage access, public file and delete file.

Figure **4.34** is the interface of a dialog that allows user to control permission on their documents. This dialog pops up when user click button "Manage access" on Figure **4.33**. The interface displays the document's name and size, ensuring users are aware of the specific document they are sharing. There is a search bar for adding members to share the document with,

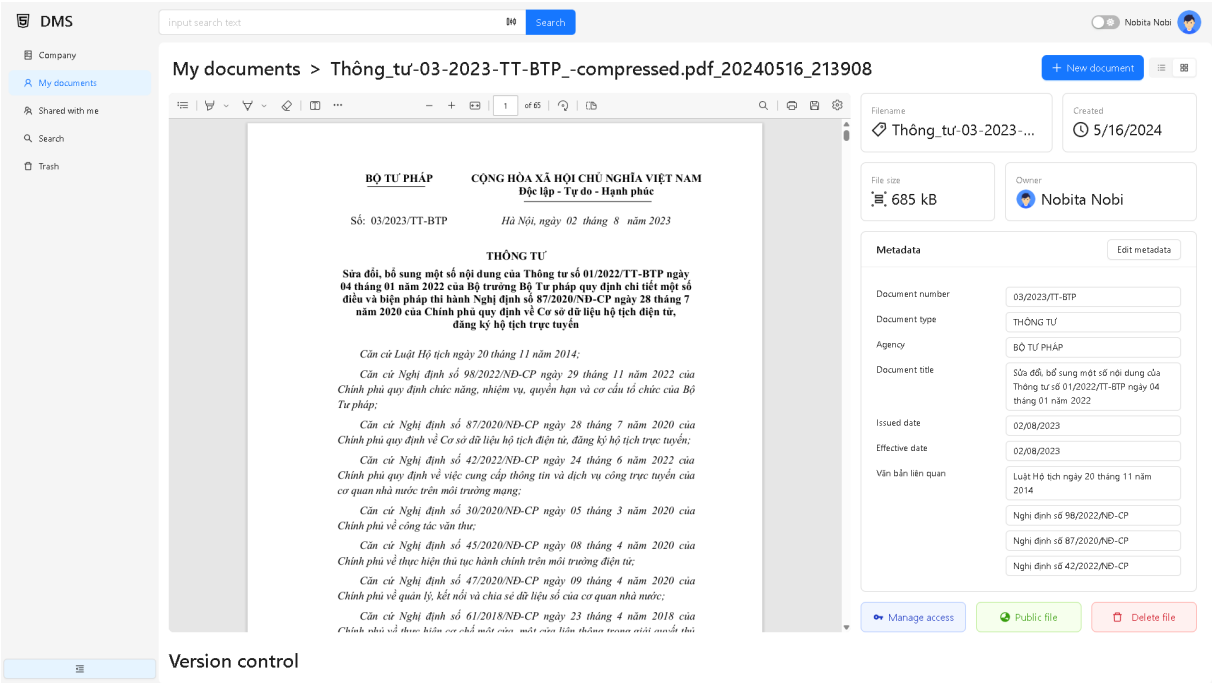

Figure 4.33: User interface - Details of a document

enhancing user experience by simplifying the process of finding and adding members.

Already shared members are listed below with their permission levels clearly indicated beside their names. Users can easily modify these permissions through drop-down menus, offering flexibility in collaboration. The options to cancel or confirm sharing actions are distinctly placed at the bottom of the dialog, promoting an efficient user experience.

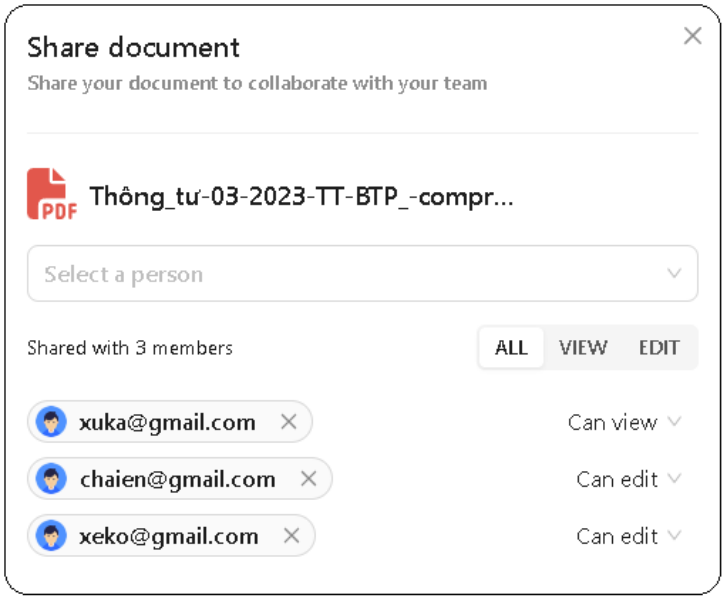

Figure 4.34: User interface - Permission modal

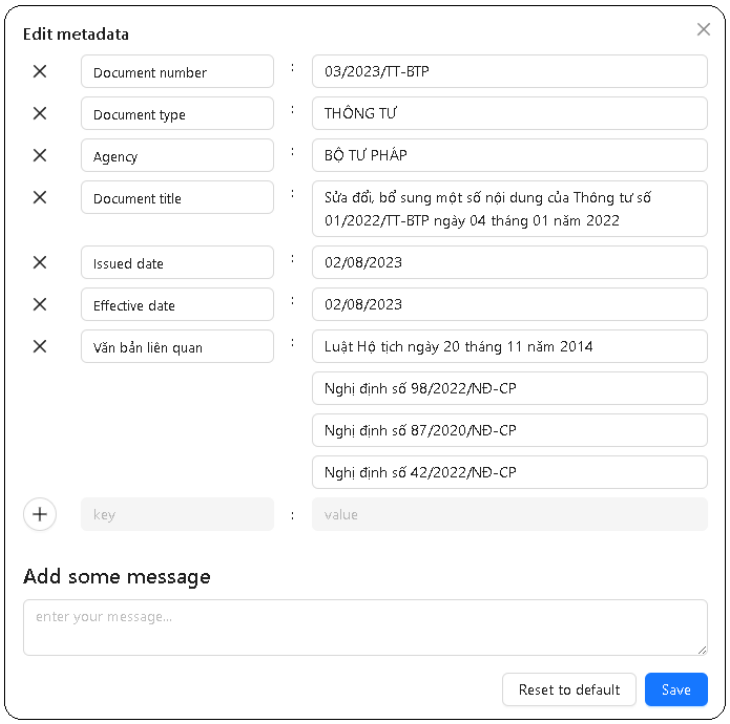

Figure 4.35: User interface - Edit metadata modal

Figure **4.35** is the interface of a dialog that allows user to edit metadata of their documents. Similar to step 3 of the uploading process (figure **4.31**), this modal allows users to freely add new metadata by entering a new pair of key - value, as well as making changes to the existed metadata. There is also a message field allowing users to explain some description for their changes, thereby improve the effectiveness of collaboration between different users on a same

document.

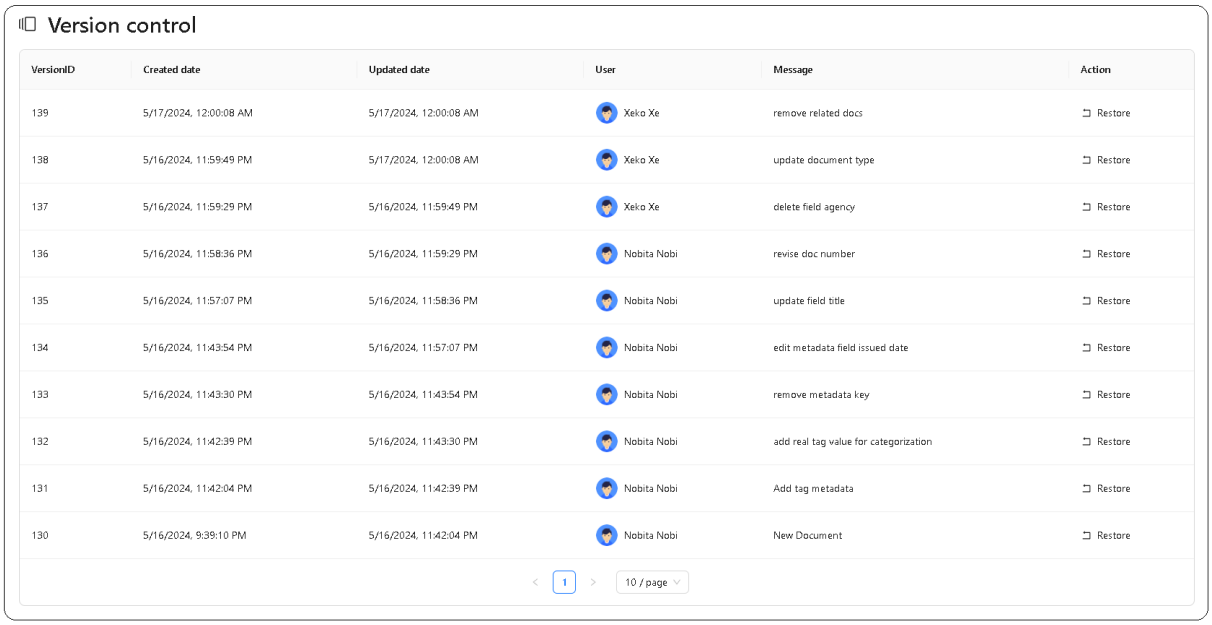

Figure 4.36: User interface - Version control

Figure **4.36** is the interface that allows users to manage different versions of their document. Each row of this table a version of the document being recorded, with information such as the time it was updated, the user who made the update and the message written by that user. Users can restore any version of the document by clicking the “Restore” button, then the state of that document will be restored back to the version that users desire.

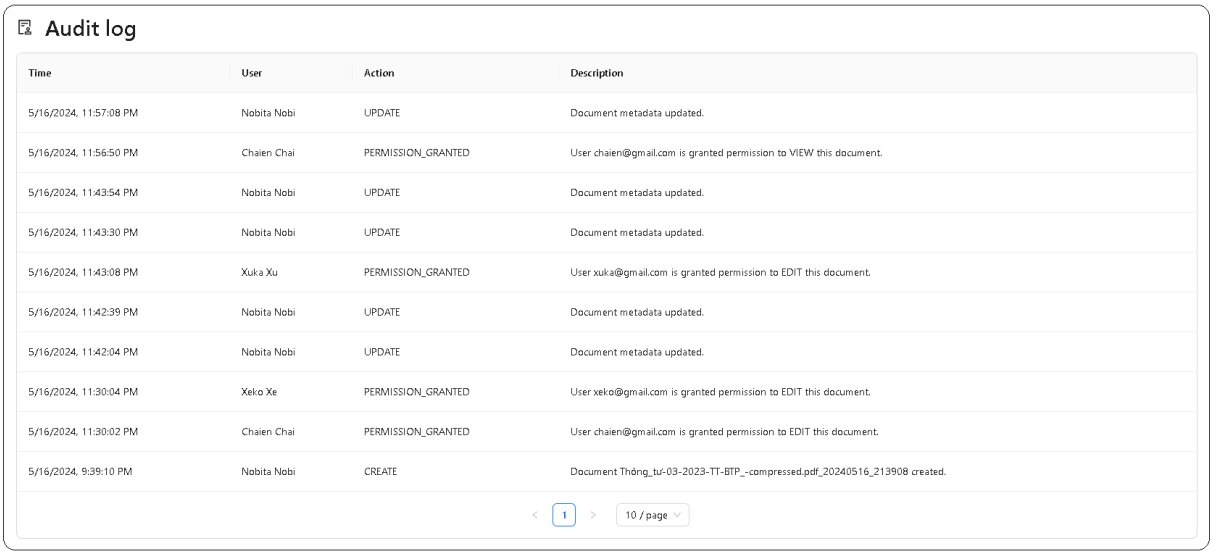

Figure 4.37: User interface - Audit log

Figure **4.37** is the interface that log actions of users on a document. Each row of this table

shows information about the time that action was conducted, the user who did the action and the type of action, such as document updated or permission granted. This table ensures that every actions on the document is logged, which can later be used for tracking and fixing if any error may occurs.

Search interface

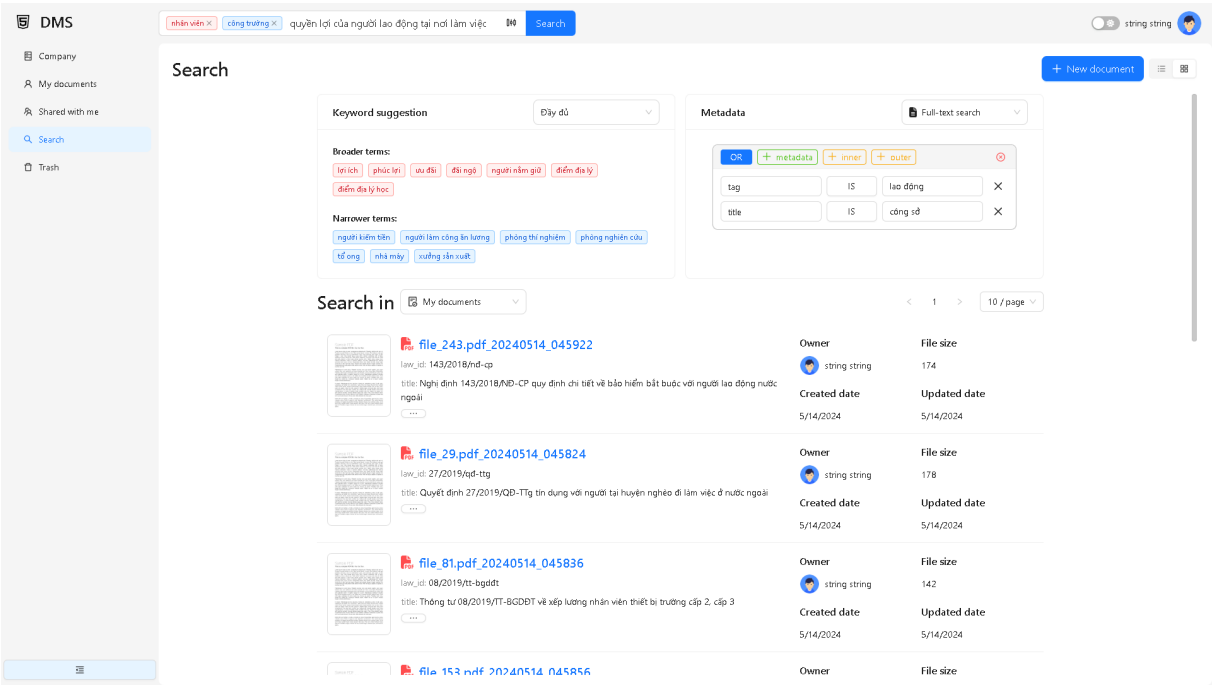

Figure 4.38: User interface - Search page

Figure **4.38** displays the search UI of our system. The searching page allows users to find documents in the system using various methods and options. Users can enter their queries in the search bar and see the results in the middle panel. The results show the document name, size, owner, created date,last modified date and the metadata of that document. Users can also navigate through multiple pages of results using the pagination controls at the bottom.

When users enter a query that matches keywords in the ontology, those keywords will be suggested to user, including broader and narrower terms that can help users refine their queries. Users can click on any of these keywords to add them to their original query and perform a query expansion.

For more advanced search options, users can use the top right panel. This panel allows users to filter the results by metadata, using many operators such as AND, OR, NOT which can be nested together to form a complex condition. Users can also choose between two different search

engines: full-text search and semantic search. Full-text search returns documents that contain the exact words or phrases in the query, along with the synonyms terms expaned by ontology. Semantic search returns documents that are semantically related to the query by comparing cosine similarity between embedding of document and user's query. Users can switch between these two modes by using the select button

Ontology interface

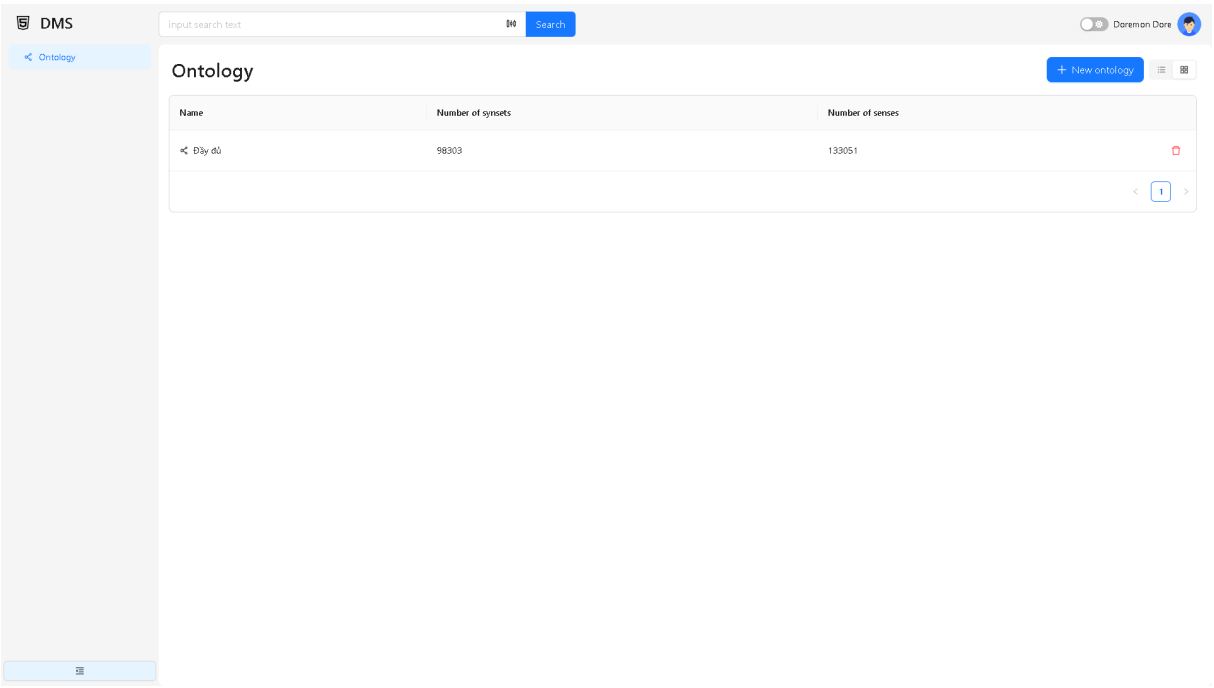

Figure 4.39: User interface - Ontology feed

Figure **4.39** is the ontology feed which displays all ontologies in the system. List of ontologies is displayed in table with three rows: the ontology name, number of synsets and number of words. Currently there is one default ontology created by us using method in Section **4.7**, expert users can create more ontologies by clicking on "New ontology" button.

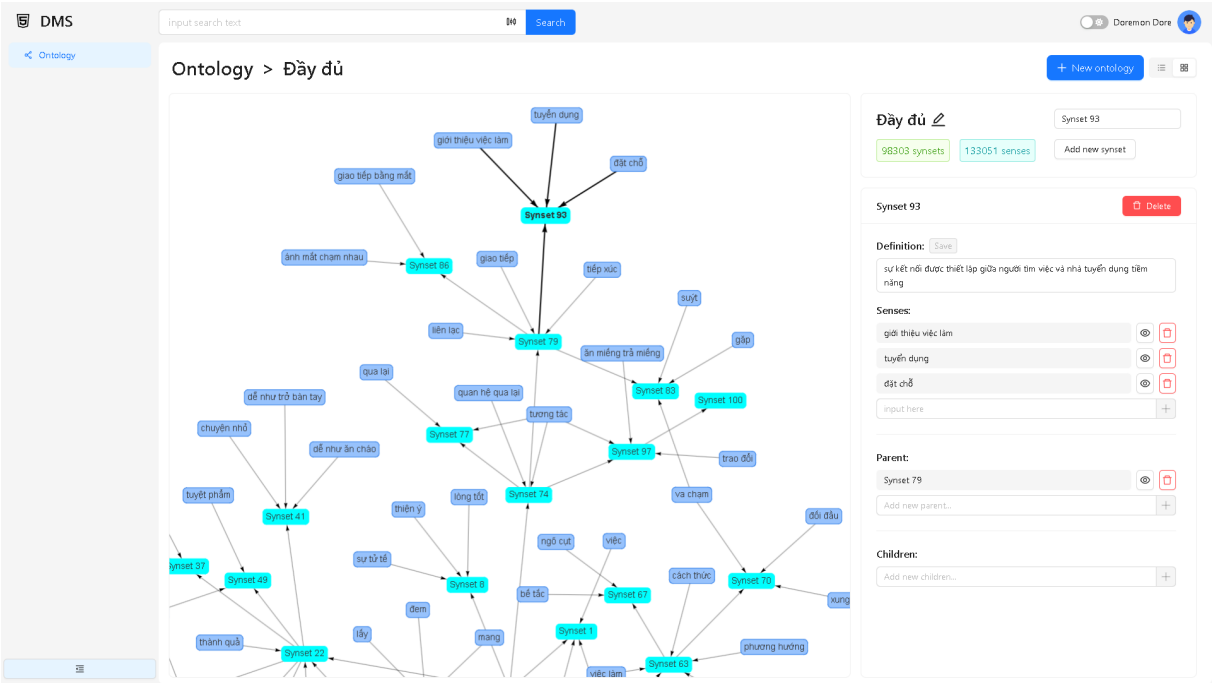

Figure 4.40: User interface - Ontology dashboard

Figure **4.40** is the ontology dashboard which renders the graph of an ontology. In this graph, synsets are the cyan node, while words are blue nodes. User can click on a synset to view its information, including "Definition", "Words" (synonyms), "Parent" (hypernyms) and "Children" (hyponyms). User can create connection between synsets and words by searching for desired synset/word in the input field and click the "Plus" button.

Chapter 5

System testing and evaluation

In this chapter, we conduct the testing for our software. Testing plays in guaranteeing the effectiveness and performance of software. By including this chapter, we acknowledges that thorough testing is a fundamental step in achieving successful software development and deployment. The inclusion of this chapter reflects the recognition that meticulous testing is indispensable for the successful creation and implementation of software solutions. Development efforts are only meaningful if the final products are functional and user-friendly.

5.1 Functional testing

In this section, we outline the procedures and results of our functional testing, focusing on verifying that each component of the software operates according to the specified requirements. Functional testing is crucial as it ensures that the system performs its intended functions correctly and reliably under specified conditions. By systematically testing individual functions, we can identify and rectify issues early in the development process, leading to a more robust and user-friendly final product.

5.1.1 Authentication and user management testing

Table 5.1: Testing – Authentication and User management

ID	Function	Test data	Expected result	Status
1	End user successful login	Email: 'string@gmail.com' Password: 'password'	User login successfully and navigate to document page	Pass
2	Expert user successful login	Email: 'expert1@gmail.com' Password: 'password'	User login successfully and navigate to ontology page	Pass

3	Wrong credentials	Email: ‘string@gmail.com’ Password: ‘pass’	System notified ”wrong credentials”	Pass
4	Create new user	Email: ‘abc@gmail.com’ Password: ‘password’	Account created successfully	Pass
5	Edit user info	Email: ’string2@gmail.com	User info updated successfully	Pass
6	Reset password	Email: ’string2@gmail.com	User info updated successfully	Pass

5.1.2 Upload document testing

Table 5.2: Testing – Upload document

ID	Function	Test data	Expected result	Status
1	Step 1 - Upload draft file	File ‘thongtu.pdf’	Draft file uploaded successfully and rendered to user	Pass
2	Step 2 - OCR content	File ‘thongtu.pdf’	Document content being extracted successfully	Pass
3	Step 3 - Extract metadata from document	File ‘thongtu.pdf’	Metadata extracted successfully	Pass
4	Edit existing metadata	Key: ‘Document type’ Value: ‘nghị định’	Metadata changed successfully	Pass
5	Delete metadata	Key: ‘Issued date’ Value: ‘2022-05-08’	Metadata deleted successfully	Pass
6	Add new metadata	Key: ‘tag’ Value: ‘giáo dục, đại học’	New metadata added successfully	Pass
7	Save document	File ‘thongtu.pdf’ with its metadata	Document saved successfully to database and cloud service	Pass

5.1.3 Document management testing

Table 5.3: Testing – Document management

ID	Function	Test data	Expected result	Status
1	Add new metadata	Key: ‘tag’ Value: ‘nông nghiệp’	Metadata added successfully	Pass
2	Edit metadata	Key: ‘Document type’ Value: ‘nghị định’	Metadata updated successfully	Pass
3	Delete metadata	Key: ‘Issued date’ Value: ‘2022-05-08’	Metadata deleted successfully	Pass
4	Create new version via update metadata	Key: ‘Issued date’ Value: ‘2022-05-08’	New version created successfully	Pass
5	Create new version via upload new file version	File ‘nghidinh.pdf’	New version created successfully	Pass
6	Revert back to old version	Version with ID ‘3’	Old version reverted successfully	Pass
7	Temporary delete document	Document with ID ‘7’	Document deleted successfully and moved to Trash	Pass
8	Restore deleted document	Document with ID ‘7’	Document restored successfully and available to users with permission	Pass
9	Delete document forever	Document with ID ‘7’	Document removed forever from database and cloud service	Pass

5.1.4 Document permission testing

Table 5.4: Testing – Document permission

ID	Function	Test data	Expected result	Status
1	Grant permission to a user	User ‘nobita@gmail.com’ with permission ’VIEW’	User can view document	Pass
2	Change’s user permission	User ‘nobita@gmail.com’ with permission ’EDIT’	User can edit document	Pass

3	Remove's user permission	User 'nobita@gmail.com'	User can no longer access document	Pass
4	Make document public	Document 'thongtu.pdf'	Document is accessible by all users in the system	Pass
5	Make document private	Document 'thongtu.pdf'	Document is private and only authorized users can access	Pass
6	Lock document	Document 'thongtu.pdf'	Document is only editable by owner, other users can only view even with EDIT permission	Pass
7	Unlock document	Document 'thongtu.pdf'	Document is editable by owner users with EDIT permission	Pass

5.1.5 Search document testing

Table 5.5: Testing – Search document

ID	Function	Test data	Expected result	Status
1	Full-text search	Query: 'quyền lợi của người lao động tại nơi làm việc' Search mode: full-text	Relevent documents are returned with set of broader and narrower terms related to user's query	Pass
2	Query expansion	Query: 'quyền lợi của người lao động tại nơi làm việc' Search mode: full-text Keywords: đãi ngộ, nhà máy	Relevent documents are returned	Pass
3	File name search	Query: 'thongtu.pdf' Search mode: filename	Documents with matching file names are returned	Pass

4	Semantic search	Query: 'quyền lợi của người lao động tại nơi làm việc' Search mode: 'semantic'	Relevent documents are returned	Pass
5	Advanced search	Query: 'quyền lợi của người lao động tại nơi làm việc' Conditions: 'tag' is 'lao động, doanh nghiệp'	Relevent documents with matching metadata conditions are returned	Pass

5.1.6 Ontology management testing

Table 5.6: Testing – Ontology management

ID	Function	Test data	Expected result	Status
1	Create new ontology	Ontology name: 'Khoa học máy tính'	New ontology created	Pass
2	Add new synset	New synset	New synset is added to the ontology	Pass
3	Edit synset definition	Definition: 'ngành khoa học kỹ thuật nghiên cứu (với sự hỗ trợ của máy tính) các quy trình và cấu trúc có thể tính toán'	Synset's definition is updated	Pass
4	Link hyponym (narrower concept) to synset	Synset ID: '23307'	Synset with ID '23307' is linked as hyponym to current synset	Pass
5	Link hypernym (broader concept) to synset	Synset ID: '23309'	Synset with ID '23309' is linked as hypernym to current synset	Pass
6	View hyponym (narrower concept) of synset	Synset ID: '23307'	The graph zoom to target synset	Pass

7	View hypernym (broader concept) of synset	Synset ID: ‘23309’	The graph zoom to target synset	Pass
8	Delete hyponym (narrower concept) of synset	Synset ID: ‘23307’	Connection between two synset is removed	Pass
9	Delete hypernym (broader concept) of synset	Synset ID: ‘23309’	Connection between two synset is removed	Pass
10	Add new word	Word: ‘học máy’	New word is added	Pass
11	Update word	Word: ‘trí tuệ nhân tạo’	Word lexeme is updated	Pass
12	Link word to synset	Synset ID: ‘22587’ Word: ‘trí tuệ nhân tạo’	Target word belongs to target synet	Pass
13	Delete synset	Synset ID: ‘22587’	Synset and its edge are deleted	Pass
14	Delete word	Word: ‘trí tuệ nhân tạo’	Word and its edge are deleted	Pass
15	Search for synset	Synset ID: ‘23309’	Graph zoom to the desired sysnet	Pass
16	Search for word	Word: ‘công nghệ phần mềm’	Graph zoom to the desired word	Pass

5.2 Application evaluation

In this section, we evaluate quality of our application website, focusing on its performance, accessibility, adherence to best practices, and search engine optimization (SEO). Utilizing the Lighthouse tool to gather objective data on various performance metrics, we aim to provide a comprehensive analysis of the website’s quality.

The result is shown in Figure **5.1**. The findings indicate that while the website excels in accessibility, best practices, and SEO, there are notable performance issues that warrant attention to enhance user experience and overall functionality.

1. **Performance**: The DMS website achieved a performance score of 82. This score, while above average, suggests there are several areas where the site can improve. Key performance metrics include:

    - **First Contentful Paint (FCP)**: Measures the time from when the page starts loading

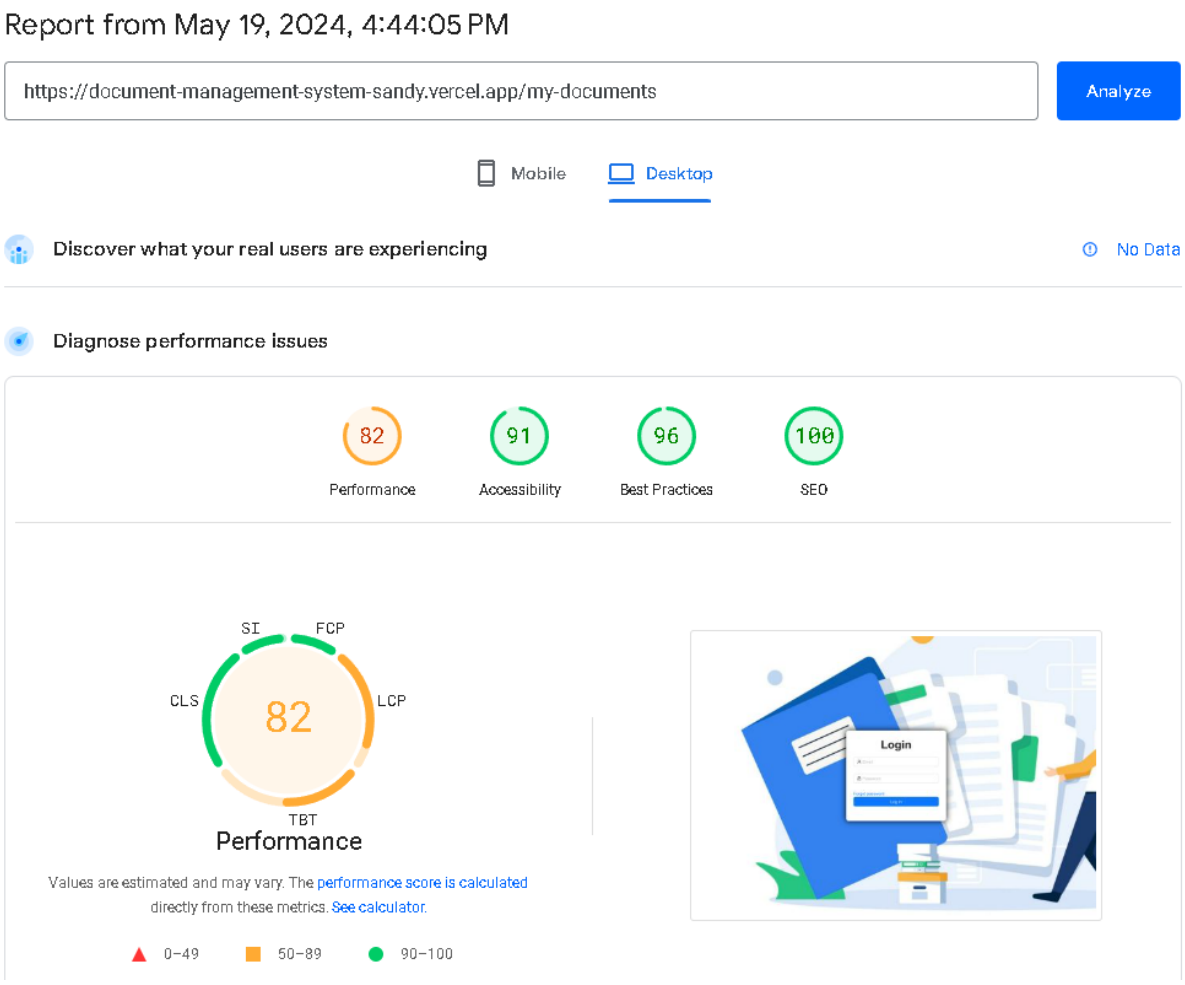

Figure 5.1: Evaluation of the website using Lighthouse

to when any part of the page's content is rendered on the screen.

- **Largest Contentful Paint (LCP)**: Marks the time at which the largest text or image is painted.
- **Total Blocking Time (TBT)**: Represents the total amount of time that the main thread was blocked and unable to respond to user input.
- **Cumulative Layout Shift (CLS)**: Quantifies how much the page layout shifts during the loading phase.

2. **Accessibility**: The accessibility score is 91, indicating the website is largely compliant with accessibility standards, allowing users with disabilities to navigate and interact with the content effectively. This high score reflects a commitment to inclusivity and usability for all users.

3. **Best Practices**: The website scored 96 in best practices, demonstrating strong adherence

to modern web development standards. This includes secure connections, responsive design, and the use of modern web technologies.

4. **SEO**: The website achieved a perfect score of 100 in SEO, indicating it is highly optimized for search engines. This ensures that the website is easily discoverable by users through search engines, which is crucial for attracting and retaining visitors.

The application shows a commendable performance in accessibility, best practices, and SEO. The high scores in these areas suggest that the application is user-friendly, secure, and optimized for search engine visibility. However, the performance score of 82 indicates room for improvement. To enhance performance, it is recommended to optimize images, leverage browser caching, and minimize render-blocking resources. Addressing these issues can reduce load times and improve the overall user experience.

5.3 Ontology Building Evaluation

To evaluate this translation, we use BLEU [27] (bilingual evaluation understudy), an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics. Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality.

Table 5.7: BLEU score interpretation [28]

BLEU score	**Interpretation**
< 0.1	Almost useless
0.1 - 0.9	Hard to get the gist
0.2 - 0.9	The gist is clear, but has significant grammatical errors
0.3 - 0.4	Understandable to good translations
0.4 - 0.5	High quality translations
0.5 - 0.6	Very high quality, adequate, and fluent translations
> 0.6	Quality often better than human

BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts.

Few human translations will attain a score of 1, since this would indicate that the candidate is identical to one of the reference translations. For this reason, it is not necessary to attain a score of 1. As from **Table 5.7**, a translation can be considered high quality if its BLEU score is over 0.4.

Mathematically, the BLEU score is defined as:

$$\text{BLEU} = \underbrace{\min\left(1, \exp\left(1 - \frac{\text{reference} - \text{length}}{\text{output} - \text{length}}\right)\right)}_{\text{brevity penalty}} \underbrace{\left(\prod_{i=1}^{4} precision_i\right)^{1/4}}_{\text{n-gram overlap}} \tag{5.1}$$

with

$$precision_i = \frac{\sum_{\text{snt} \in \text{Cand} - \text{Corpus}} \sum_{i \in \text{snt}} \min\left(m^i_{cand}, m^i_{ref}\right)}{w^i_t = \sum_{\text{snt}' \in \text{Cand} - \text{Corpus}} \sum_{i' \in \text{snt}'}, m^{i'}_{cand}} \tag{5.2}$$

where

- m^i_{cand} is the count of i-gram in candidate matching the reference translation
- m^i_{ref} is the count of i-gram in the reference translation
- w^i_t is the total number of i-grams in candidate translation

The formula consists of two parts: the brevity penalty and the n-gram overlap.

- **Brevity Penalty** The brevity penalty penalizes generated translations that are too short compared to the closest reference length with an exponential decay. The brevity penalty compensates for the fact that the BLEU score has no recall term.
- **N-Gram Overlap** The n-gram overlap counts how many unigrams, bigrams, trigrams, and four-grams (i=1,...,4) match their n-gram counterpart in the reference translations. This term acts as a precision metric. Unigrams account for adequacy while longer n-grams account for fluency of the translation. To avoid overcounting, the n-gram counts are clipped to the maximal n-gram count occurring in the reference (m^n_{ref}).

Given the large scale of the ontology comprising 98,303 synsets, a comprehensive evaluation of translation quality across all synsets is impossible. Consequently, a sampling approach becomes neccessary to evaluate the quality of the entire ontology. Various sampling methods exist, including simple random sampling, systematic sampling, and cluster sampling,... In this study, we use stratified sampling method to ensure that the selected subset is as representative of the ontology as possible. In the original Princeton Wordnet, some synsets are labeled by the

Table 5.8: Ontology translation evaluation

Synset domain	**BLEU Score**
Pháp luật (Law)	0.40
Khoa học máy tính (Computer science)	0.57
Thương mại (Commerce)	0.41
Hình sự (Crime)	0.39
Kinh tế (Economics)	0.63

"Domain" attribute, which denotes the specialized domain to which a synset belongs. Therefore, by leveraging this "Domain" attribute, we categorize synsets into distinct subgroups, each corresponding to a specialized domain. For evaluation purposes, five domain groups are selected: *Pháp luật (Law)*, *Khoa học máy tính (Computer Science)*, *Thương mại (Commerce)*, *Hình sự (Crime)*, *Kinh tế (Economics)*. Then, twenty synsets are randomly chosen from each group, resulting in a total of 100 synsets. The evaluation metric BLEU score is then calculated for each synset within each group to assess translation quality. The results of this evaluation are presented in **Table 5.8**.

5.4 Query Ontology Evaluation

The process of query expansion utilizes keywords retrieved from the ontology, thereby demonstrating the ontology's efficacy when these keywords are semantically equivalent to the original terms from the user's query. This semantic equivalence suggests that the retrieved keywords can be used interchangeably with the original terms in certain contexts. The performance of the ontology retrieval process is quantified by the precision metric, which is defined as the ratio of precise terms relative to the total number of terms returned by the ontology, as depicted in formula 5.3 and 5.4.

$$\text{precision}_{syn} = \frac{K_{syn}}{N_{syn}} \tag{5.3}$$

$$\text{precision}_{sug} = \frac{K_{sug}}{N_{sug}} \tag{5.4}$$

This precision metric is calculated for two distinct sets of terms: synonyms (related terms) and suggestions (broader/narrower terms). Since synonyms are automatically integrated into the user's query, they require a higher level of precision compared to suggestion terms, which

Table 5.9: Querying ontology evaluation

User's query	N_{syn}	K_{syn}	N_{sug}	K_{sug}	$prec_{syn}$	$prec_{sug}$
quyền được bàn của người lao động tại nơi làm việc	6	6	25	6	1	0.64
luật hình sự các yếu tố cấu thành tội phạm	3	3	14	8	1	0.57
hình phạt đối với tội tham ô tài sản	6	6	17	12	1	0.7
các trường hợp sa thải bất hợp pháp trong luật lao động	3	3	7	7	1	1
chế độ nghỉ phép năm của người lao động	2	2	13	9	1	0.69
thủ tục chuyển nhượng quyền sử dụng đất	4	3	19	13	0.75	0.68
thủ tục thành lập doanh nghiệp	3	2	18	10	0.66	0.56
các quy định về quyền sở hữu trí tuệ	2	2	9	6	1	0.66

are presented to the user as optional enhancements for query expansion. The results, shown in Table **5.9**, include a collection of input queries used for evaluation. These results reveal that the precision of the retrieved synonyms is consistently high for nearly all queries, indicating that the synonyms retrieved are contextually very similar to the user's original keywords. Conversely, the precision for suggestion terms is lower. This lower precision is considered acceptable as suggestion terms are optionally chosen by the user for query expansion and do not negatively impact the overall search performance.

5.5 Search Method Evaluation

5.5.1 Dataset Overview

We employ the Vietnamese legal dataset from the Zalo AI Competition 2021 to assess our system's search functionality performance. This dataset comprises legal documents organized into passages, accompanied by a training dataset containing individual questions and their corresponding answers, some of which may have multiple responses. A question in the dataset is accompanied by the relevant law document and its passages, as presented in Figure 5.2. However, in our evaluation, we focus solely on assessing the effectiveness of our document retrieval. Therefore, we consider only the law documents, not the individual passages.

```
{
    "question_id": "fc987d9e1f58a16d9b000305381ac884",
    "question": "Phá dỡ thiết bị công trình biên giới bị phạt thế nào?",
    "relevant_articles": [
        {
            "law_id": "96/2020/nđ-cp",
            "article_id": "10"
        }
    ]
},
```


Figure 5.2: Example Question

As only the training dataset is publicly available from the Zalo AI Competition, we can only consider 3196 question-answer in this set. Upon analysis, it was observed that out of the total 3109 documents, only 705 were represented in the question-answer dataset. Among these 705 documents, approximately 36% of them appeared only once in the question-answer dataset.

As a result, due to hardware limitation, instead of ingesting over 3000 documents into Elasticsearch, we only chose those 705 documents that appeared in the question-answer dataset.

5.5.2 Search method settings

- **Full-text Search**: Scored documents using BM25.
- **Query Expansion with Domain Ontology**: Scored documents using BM25. Utilized Legal Domain Ontology to expand query, where related terms are identified and automatically expanded with Boolean operator OR.
- **Semantic search**: Content chunk size is 600 tokens with a chunk overlap of 100 tokens. Utilized the ”text-embedding-3-large” model from OpenAI for creating embedding, the embedding vector has a dimension of 768. Semantic similarity is measured using Cosine Similarity.

5.5.3 Evaluation Metrics

A notable characteristic of the Zalo legal question-answering dataset is that among the 3196 questions, a vast majority, 3103 in total, have only one positive document associated with them, constituting over 97% of the dataset. The remaining 3% of questions possess two or three positive passages, indicating a relatively small proportion.

Due to this imbalance, rather than employing the conventional recall score utilized in many information retrieval studies, we opt for accuracy to evaluate the performance. Accuracy is measured as the proportion of samples where at least one relevant document is returned within

the top k passages scored by the models. The formula for this measure is as follows:

$$acc_k = \frac{n_{p,k}}{n}.100\%$$

where n is the number of samples in the evaluating dataset, and $n_{p,k}$ is the number of samples having at least one relevant documents retrieved in the top-k documents.

While Accuracy@K provides valuable insights into the retrieval performance, it does not account for the order in which the relevant documents are presented. To address this, we want to incorporating the **Normalized Discounted Cumulative Gain (NDCG@K)** as an additional evaluation metric. NDCG@K evaluates how well a search method ranks the relevant documents by considering the relative order of the returned items. This metric is particularly important because users typically examine only the top few search results, making the order of the results more significant than their absolute count. The introduction of NDCG@K allows us to better understand the effectiveness of each search methods in ranking relevant documents.

$$\text{NDCG@k} = \frac{DCG@k}{IDCG@k} \tag{5.5}$$

where,

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{5.6}$$

$$IDCG@k = \sum_{i=1}^{min(k,|REL|)} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{5.7}$$

- NDCG@k is the Normalized Discounted Cumulative Gain at rank k.

- $DCG@k$ is the Discounted Cumulative Gain at rank k.

- $IDCG@k$ is the Ideal Discounted Cumulative Gain at rank k.

- rel_i is the relevance of the item at rank i (relevant document is 1 and if not, it's 0).

- $|REL|$ is the total number of relevant items.

5.5.4 Results

We evaluate the search performance using top 1, 5, 10, 20, 30 retrieved documents. The results is displayed in Table 5.12 and 5.11

Search Method	**Accuracy@K**				
	1	5	10	20	30
Full-Text Search(%)	53.72	76.83	84.43	90.52	93.58
Full-Text Search + Query Expansion (%)	55.35	79.08	85.99	91.73	94.21
Semantic Search (%)	80.78	96.43	98.44	99.22	99.53

Table 5.10: Comparison of Search Methods using Accuracy@K

Search Method	**NDCG@K**				
	1	5	10	20	30
Full-Text Search + Query Expansion	0.5372	0.6628	0.6872	0.7024	0.709
Full-Text Search + Query Expansion	0.5535	0.6826	0.7049	0.7193	0.7245
Semantic Search	0.8076	0.8971	0.9036	0.9053	0.9058

Table 5.11: Comparison of Search Methods using NDCG@K

From the comparison of the search methods in **Table 5.12 and 5.11**, several observations can be made:

1. **Full-Text Search vs. Full-Text Search + Query Expansion**:

   Both methods utilize the BM25 algorithm for scoring documents, but the latter incorporates a domain-specific ontology to expand the query terms, aiming to enhance retrieval performance by including related terms. The results indicate that this expansion indeed provides a modest improvement:

   - **Accuracy@K**: Full-text search achieves an accuracy of 53.72% at K=1, which improves to 55.35% with query expansion. This trend of improvement continues across all evaluated K values, suggesting that the inclusion of related terms helps in retrieving more relevant documents. By K=30, the accuracy rises to 93.58% for full-text search and 94.21% for query expansion.

   - **NDCG@K**: The normalized discounted cumulative gain (NDCG), which considers the rank order of relevant documents, also shows improvement. For instance, NDCG@1 increases from 0.5372 to 0.5535, and at K=30, it goes from 0.709 to 0.7245. This indicates that not only are more relevant documents being retrieved with query expansion, but they are also ranked higher.

These metrics collectively suggest that query expansion makes the search more effective, likely because it captures a broader range of relevant terminology, improving both the quantity and the ranking of relevant documents retrieved.

2. **Full-Text Based Methods vs. Semantic Search**

   Semantic search, which uses embeddings and measures similarity via cosine similarity, vastly outperforms both full-text methods:

   - **Accuracy@K**: At K=1, semantic search achieves an impressive 80.78% accuracy, far exceeding the 55.35% of query expansion and 53.72% of standard full-text search. This substantial lead is maintained across all K values, with semantic search reaching 99.53% accuracy at K=30.
   - **NDCG@K**: The NDCG metrics similarly reflect this superiority. At K=1, semantic search achieves an NDCG of 0.8076, significantly higher than the 0.5535 of query expansion and 0.5372 of full-text search. Even at K=30, semantic search maintains a high NDCG of 0.9058, indicating that it consistently ranks relevant documents higher than the other methods.

5.5.5 Heuristic Analysis

Dataset Drawbacks

The evaluation dataset has certain limitations that impact the assessment of the search methods. Over 97% of the questions in the dataset are associated with only one relevant document. This skewed distribution means the evaluation metrics, particularly accuracy and NDCG, are influenced by a binary relevance judgment (1 for relevant, 0 for not), which does not fully capture the potential of the semantic search method and query expansion method.

Impact of Dataset Limitations

- **Binary Relevance Limitation**: The binary nature of the relevance judgment in the dataset restricts the ability to measure the effectiveness of the search methods comprehensively. In reality, documents can have varying degrees of relevance, but this nuance is lost in a binary evaluation system. NDCG, which is sensitive to the ranking of relevant documents, may not reflect the true effectiveness of search method if only one document is marked as relevant per query.
- **Single Relevant Document**: Given that most questions have only one marked relevant document, the benefits of semantic search, which excels at understanding the broader con-

text and identifying multiple relevant documents, are not fully demonstrated. Semantic search can retrieve documents that are contextually related or cover the same topic, providing a richer set of relevant results that a binary relevance metric fails to acknowledge.

Despite these limitations, qualitative analysis of the results indicates that both search methods offer unique performance:

Semantic Search

- **Contextual Relevance**: Semantic search, by leveraging embeddings and cosine similarity, captures the context and nuances of legal terminology better than full-text methods. This results in retrieving documents that are contextually related to the query, even if they are not marked as the single relevant document in the dataset.
- **Topic Coherence**: The result sets returned by semantic search are often topically coherent, meaning they include documents that, while not labeled as relevant, are related to the same subject matter or legal issue as the query. This suggests a higher level of understanding and relevance that is not fully captured by the current evaluation metrics.

Full-Text Method with Query Expansion:

- **Domain-specific Enhancement**: Query expansion based on legal domain ontology enriches the search process by incorporating related terms, bridging the vocabulary gap between queries and documents in the legal domain. This can lead to the retrieval of more relevant documents that cover a wider range of legal topics.
- **Broadened Query Scope**: By broadening the scope of the queries, query expansion increases the likelihood of retrieving documents that are contextually related, even if they are not explicitly marked as relevant in the dataset.

5.5.6 Conclusion

Incorporating query expansion using domain-specific ontology into full-text search improves retrieval performance, but the most significant gains are observed when employing semantic search techniques. The ability of semantic search to understand and interpret the meaning of queries and documents results in markedly higher accuracy and better ranking of relevant documents. This demonstrates that leveraging semantic technologies can greatly enhance the effectiveness of document retrieval components, especially in specialized fields like legal information retrieval.

The dataset's constraints, specifically the prevalence of single relevant document associa-

tions and binary relevance scoring, limit the ability to fully showcase the strengths of semantic search and query expansion. While traditional metrics like accuracy and NDCG provide a snapshot of performance, they do not account for the nuanced relevance and contextual understanding that both methods bring.

Qualitative observations suggest that semantic search retrieves more contextually relevant and topically coherent documents, indicating its potential superiority in real-world applications where relevance is not binary. Future evaluations would benefit from a more detailed relevance scoring system and a dataset with multiple relevant documents per query to better assess and highlight the strengths of both search methods: Semantic search and Query expansion.

5.6 Metadata Extraction Evaluation

In this section, for evaluating the Metadata Extraction module, we want to assess how accurate the process flow from OCR the document to extract metadata value from the OCR-processed document.

5.6.1 Dataset Overview

The dataset utilized in this study aligns with the same dataset employed in the Search evaluation, comprising approximately 705 legal documents. However, post-processing, aimed at obtaining the PDF files corresponding to each legal document from the https://luatvietnam.vn/ website, revealed that not all documents were accompanied by a PDF version. Consequently, a decision was made to exclude documents lacking a PDF version, resulting in a refined dataset comprising 636 documents.

Subsequently, for each PDF file retrieved from the website, Optical Character Recognition (OCR) was conducted to extract the content of each document. It's noteworthy that OCR was systematically applied to all documents, even those with pre-existing OCR content. This extracted content serves as the input for the metadata extraction process.

Moreover, leveraging the Vietnamese legal dataset from the Zalo AI Competition 2021, supplementary metadata is available for each document. The provided metadata includes Law ID, Document Type, and Issued Department, as illustrated in Figure 5.3. Consequently, the evaluation will focus on assessing the performance of our Metadata Extraction module in accurately extracting the three metadata fields: Law ID, Document Type, and Issued Department, utilizing the corresponding data from the Zalo Legal Dataset as the ground truth.

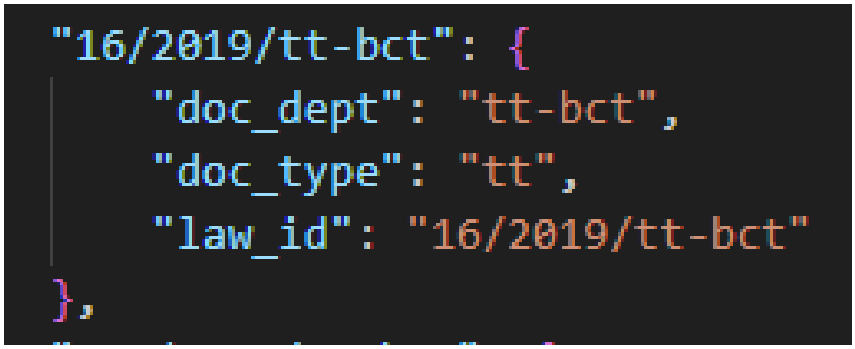

Figure 5.3: Example Metadata

5.6.2 Metadata Extraction module settings

In the evaluation, we will prompt the OpenAI model gpt-3.5-turbo-0125 together with prompt instructions to extract the metadata value from the document, the setting for prompt is as presented in Figure 5.4, the temperature is set low as 0.1 to ensure the consistency in the returned format:

```
model="gpt-3.5-turbo-0125",
messages=[
    {
    "role": "system",
    "content": [
        {
        "type": "text",
        "text": "Extract the possible document metadata from the legal document text
        provided (this is the first page of the document), return the result in Vietnamese
        with the JSON format \"metadata\":\"value\", correct any grammatical errors, this
        is the set of metadata with its type hinting in python, it must strictly follow
        this:\n\"Loại văn bản\": (type: str)\n\"Số\": (type: str)\n\"Đơn vị ban hành\":
        (type: str)"
        }
    ]
    },
    {
    "role": "user",
    "content": [
        {
        "type": "text",
        "text": content
        }
    ]
    },
    {
    "role": "assistant",
    "content": [
        {
        "type": "text",
        "text": "{\n    \"Loại văn bản\": \"\",\n    \"Số\": \"\",\n    \"Đơn vị ban
        hành\": \"\"}"
        }
    ]
    }
],
temperature=0.1,
max_tokens=1024,
top_p=1,
frequency_penalty=0,
presence_penalty=0
```

Figure 5.4: Prompt setting

5.6.3 Evaluation Metrics

We utilized accuracy as the metric to assess the effectiveness of the metadata extraction method for individual metadata fields. Accuracy is defined as the ratio of correctly extracted metadata samples to the total number of samples in the evaluating dataset, expressed as a percentage. The formula for this measure is as follows:

$$acc_k = \frac{n_{p,k}}{n}.100\%$$

Where n represents the total number of samples in the evaluating dataset, and $n_{p,k}$ denotes the number of samples where the extracted metadata matches the corresponding ground truth value.

5.6.4 Results

We assess the performance of metadata extraction across three fields: Law ID, Document Type, and Issued Department. The results are presented in Table 5.12

Metadata Field	Accuracy (%)
Law ID	62.26
Document Type	91.04
Issued Department	63.84

Table 5.12: Accuracy of metadata extraction for each field.

The evaluation of the Metadata Extraction module demonstrated varying degrees of accuracy across different metadata fields. Specifically, the accuracy for extracting the Document Type was notably high at 91.04%, indicating a robust performance in identifying this field accurately. Conversely, the accuracy for extracting the Law ID and Issued Department fields was lower, at 62.26% and 63.84% respectively.

These results highlight the strengths and limitations of the current OCR and metadata extraction process. The high accuracy for Document Type suggests that this metadata field is well-defined and consistently recognized by the extraction module. However, the comparatively lower accuracy for Law ID and Issued Department indicates that further improvements are needed to enhance the reliability of the extraction process for these fields.

The noise produced by OCR, including misrecognition and formatting issues, likely contributes to the reduced accuracy in certain fields. This suggests that the OCR process may require further refinement to minimize noise and improve the reliability of text recognition, particularly for complex documents and less consistent metadata fields.

Overall, while the module shows promise, especially for certain types of metadata, additional refinement and optimization are necessary to achieve more uniform performance across all metadata fields. Future work may involve fine-tuning the OCR process, enhancing the prompt settings, and incorporating more sophisticated extraction techniques to improve the overall accuracy and robustness of the metadata extraction module.

Chapter 6

System Deployment

Deployment is a critical phase in the life-cycle of any software project, serving as the bridge from development to practical application. It transforms theoretical solutions into operational systems that directly impact end-users. In the context of our digital document management system, effective deployment ensures that the platform can manage documents efficiently and securely, supporting the needs of users in real-time environments. Our deployment strategy leverages state-of-the-art technologies and robust methodologies to ensure that the system is not only deployed efficiently but also operates reliably under varying conditions.

In this chapter, we details our comprehensive approach to deploying a complex web application using Docker containers on Amazon EC2, illustrating how we translate our software engineering efforts into a tangible and functional system that meets rigorous academic and professional standards.

6.1 Main server deployment

6.1.1 Technology Choices and Justification

The selection of technologies for our digital document management system was driven by the need for robustness, scalability, security, and cost efficiency. Here's a deeper look into why we chose each technology:

- **Django:** As a comprehensive high-level Python web framework, Django facilitates rapid development, clean, pragmatic design, and emphasizes the DRY (Don't Repeat Yourself) principle. It offers an extensive suite of built-in tools for everything from user authentication to content administration. Its built-in security features protect against many vulnerabilities by default, which is paramount for our document management system.
- **PostgreSQL:** Known for its strong standards compliance, robustness, and performance, PostgreSQL offers advanced features such as full ACID compliance for transactions, robust locking, and MVCC (Multi-Version Concurrency Control). These features make it

highly suitable for applications requiring concurrent data access, which is expected in our project.

- **Docker:** The use of Docker provides a standardized unit of software, packaging up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Docker simplifies deployment and eliminates the ”it works on my machine” problem by providing a consistent environment for development, staging, and production.
- **Amazon EC2:** Amazon EC2 provides resizable compute capacity in the cloud and makes web-scale computing easier. We chose EC2 because it integrates well with other AWS services, provides a secure, resizable compute capacity in the cloud that lets you scale, and it’s highly reliable.
- **Nginx:** As a high-performance HTTP server and reverse proxy, Nginx excels at large scale web traffic handling. It serves static content, manages SSL/TLS termination, and provides HTTP caching, which offloads these tasks from the Django application, allowing it to handle dynamic content more efficiently.

6.1.2 Server Configuration

The configuration of our Amazon EC2 instance was meticulously planned to ensure optimal performance and security:

- **Instance Type: ’t2.micro’** was chosen not only for its cost-effectiveness under the AWS free tier but also for its capability to handle our current workload of development and testing with the potential to scale up as needed.
- **Operating System:** We opted for Amazon Linux 2 due to its seamless integration with AWS services and robust security. The OS is lightweight, secure, and provides a stable platform for hosting our Docker containers.
- **Security Setup:** We configured the EC2 security groups to create a stringent security environment. This setup involves defining rules that allow traffic only on necessary ports such as HTTP (80) and HTTPS (443) for web access, and SSH (22) for secure remote management, effectively guarding against unauthorized access.
- **Storage Strategy:** We use Elastic Block Store (EBS) for database and application data storage. The choice of EBS ensures data persistence across instance restarts and terminations. Regular snapshots of the EBS volumes are taken to ensure data redundancy and recovery in case of failures.

6.1.3 Server deployment process

Our deployment leverages Docker Compose to orchestrate the multi-container setup, described in our ***'docker-compose-deploy.yml'*** file. This approach not only simplifies the management of containers but also ensures consistency across environments:

```yaml
version: "3.9"

services:
  app:
    build:
      context: .
    restart: always
    volumes:
      - static-data:/vol/web
    environment:
      - DB_HOST=db
      - DB_NAME=${DB_NAME}
      - DB_USER=${DB_USER}
      - DB_PASS=${DB_PASS}
      - SECRET_KEY=${SECRET_KEY}
      - ALLOWED_HOSTS=${ALLOWED_HOSTS}
    depends_on:
      - db

  db:
    image: postgres:13-alpine
    restart: always
    volumes:
      - postgres-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=${DB_NAME}
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_PASSWORD=${DB_PASS}

  proxy:
    build:
      context: ./proxy
    restart: always
    depends_on:
      - app
    ports:
      - 80:8000
    volumes:
      - static-data:/vol/static

volumes:
  postgres-data:
  static-data:
```

a. Application Service ('app')

- **Build Context:** Specifies the directory containing the Dockerfile for building the Django application image.
- **Restart Policy:** Configured to always restart, ensuring high availability by automatically restarting the container if it crashes or the server reboots.
- **Volumes:** Utilizes a named volume static-data to persist static files across container restarts and deployments.
- **Environment Variables:** Includes connections to the database and critical Django settings such as the secret key and allowed hosts, which are loaded from environment variables to enhance security.
- **Dependencies:** Declares a dependency on the **db** service, ensuring that the database service is started before the application.

b. Database Service ('db')

- **Image:** Uses the official **'postgres:13-alpine'** image for a lightweight and efficient PostgreSQL database server.
- **Restart Policy:** Also set to always restart, enhancing the resilience of the database service.
- **Volumes:** A named volume **'postgres-data'** is used to persist database data, protecting data integrity across container rebuilds.
- **Environment Variables:** Configured with database credentials and settings, ensuring the database is initialized with the required user and database details.

c. Proxy Service ('proxy')

- **Build Context:** Points to a directory containing a Dockerfile for the Nginx proxy setup.
- **Restart Policy:** Always restart to maintain continuous availability of the proxy service.
- **Dependencies:** Set to depend on the **'app'** service, ensuring that traffic is only routed when the application is ready.
- **Port:** Maps port 80 on the host to port 8000 on the container, directing external HTTP traffic to the Django application through the proxy.

- **Volumes:** Shares the **'static-data'** volume with the application container to serve static files efficiently.

d. Volumes

These named volumes are crucial for data persistence. **'postgres-data'** keeps the PostgreSQL data safe during container restarts and redeployments, while **'static-data'** holds Django static files like CSS, JavaScript, and images. This Docker Compose configuration is key to our deployment strategy, ensuring each component is appropriately isolated, scalable, and maintainable. It also highlights our use of environment variables and volumes to secure sensitive information and maintain data persistence.

6.2 Front-end deployment

The deployment process of web applications is a critical phase in the software development lifecycle, ensuring that the application is accessible to users over the internet. In this project, the deployment of our React application system was facilitated using Vercel, a modern cloud platform known for its efficiency and ease of use. Figure **6.1** shows the result of deployment.

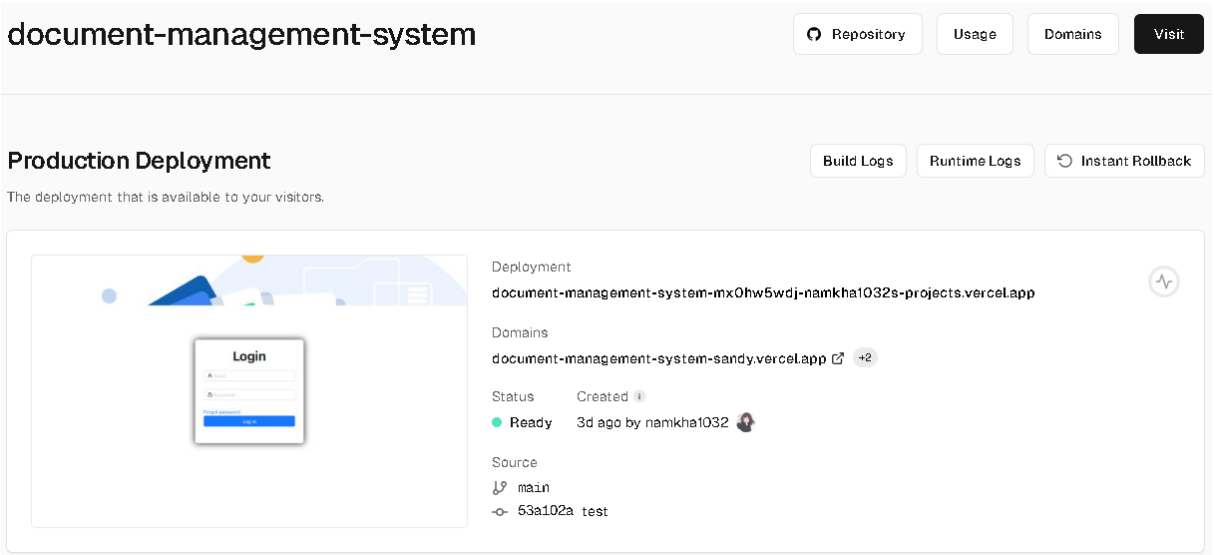

Figure 6.1: Deployment result of front-end

The deployment of the our system on Vercel involves the following steps:

1. **Repository Integration**: The first step is to integrate the application's repository with Vercel. This is accomplished by connecting the Vercel account to the GitHub repository where the application is hosted.

2. **Project Configurationn**: Upon linking the repository, Vercel automatically detects the React project and suggests default settings. These settings can be customized based on specific project requirements. Configuration options include specifying build commands, environment variables, and deployment settings. Our application was built using `create-react-app`, so configuration options are:

    - **Build command**: `react-scripts build`
    - **Output directory**: `build`
    - **Install command**: `npm install`
    - **Development commant**: `react-scripts start`

3. **Build and Deployment**: Once the project is configured, Vercel triggers the initial build process. Vercel uses its built-in infrastructure to compile the React application, optimizing it for production. The build output is then deployed to Vercel's edge network.

4. **Continuous Deployment**: Vercel supports continuous deployment, meaning any subsequent push to the repository triggers an automatic rebuild and redeployment of the application. This ensures that the latest code changes are always reflected in the deployed version.

5. **Monitoring and Management**: Post-deployment, Vercel provides tools for monitoring application performance and managing deployments. Features such as custom domains, SSL certificates, and analytics are available to ensure the application meets operational requirements.

Vercel proved to be a robust and efficient platform for deploying the React application, aligning with the project's goals of simplicity, performance, and scalability. The deployment process demonstrated how modern cloud platforms can facilitate the rapid and reliable delivery of web applications.

Chapter 7

Further improvement and Conclusion

In this chapter, we will conduct an in-depth exploration of avenues for further improvement within the context of this thesis. This discussion will encompass a detailed examination of potential enhancements, refining aspects of the research to bolster its overall depth and effectiveness. Throughout the thesis, some improvements were briefly mentioned when discussing various functionalities of the system. Here, we will revisit those improvements and introduce new potential enhancements.

7.1 Further improvement

7.1.1 Stress testing

The system was built on a local computer and hasn't dealt with a lot of users yet. So, we don't know if it can handle a lot of people using it at the same time. If we had more time and resources, we could have tested it with a lot of simulated users to see if it can handle real-world use. This testing is needed to make sure the system is ready for many users.

7.1.2 Building ontology

Due to lack of Vietnamese resources about ontology or wordnet, we have to implement the method of translating original English wordnet into Vietnamese to use as our ontology. Although the quality of this translation is proven to have high quality, there are still differences in nuance between English and Vietnamese language that this translation cannot express. The quality of this ontology can be improved with the help of lexicographer or expertise in some specific domain.

7.1.3 Querying the ontology

The execution time for querying the ontology is notably high, averaging approximately three seconds per query. This latency can be attributed to two primary factors. First is the hardware

limitations of Aura - the Neo4j cloud service utilized for ontology storage. Aura's current configuration, which includes only one CPU and 1 GB of memory, struggles to handle queries that scan over the ontology which consists of more than 100,000 nodes, resulting in a complete utilization of the available hardware resources. Addressing this challenge could involve upgrading to a higher-tier pricing plan that offers enhanced hardware capabilities or deploying the Neo4j database locally. However, due to time constraints, reliance on the existing cloud service remains necessary.

The second factor involves the inherent characteristics of Neo4j's CYPHER query language, which extensively uses cross product matching. Improper handling of these cross product matchings can lead to substantial query execution times. Given additional time, further research into the CYPHER query language could facilitate optimization efforts, thereby improving execution efficiency.

7.1.4 Search result

Our evaluation of search performance reveals that both Full-Text and Semantic Search methods exhibit strengths in retrieving relevant documents, with notable distinctions.

Full-Text + Expand consistently outperforms the baseline Full-Text approach, particularly at higher top K values, highlighting the effectiveness of query expansion in enhancing retrieval accuracy by incorporating additional relevant terms. However, Semantic Search consistently outperforms Full-Text methods by utilizing deep learning models to understand and capture the semantic meaning of queries. This approach allows for the retrieval of documents that are contextually related to the query, even if they do not contain exact keyword matches, thereby providing users with a more comprehensive set of relevant documents.

Despite these promising outcomes, our study identifies areas for improvement. We did not evaluate the semantic similarity among retrieved documents, which could further emphasize the power of semantic search in delivering contextually aligned results. Additionally, the lack of an appropriate Vietnamese document dataset limited our ability to fully evaluate the ranking and relevance of retrieved documents for each search method in this language context.

Future work should focus on integrating the strengths of both keyword-matching and semantic similarity methods to optimize document retrieval. By combining these approaches, we can enhance the retrieval of documents that not only contain the desired keywords but also match the context of user queries, ultimately improving the precision and comprehensiveness of search results.

7.1.5 Metadata Extraction

To enhance the performance of the metadata extraction module, two key areas need attention: improving OCR accuracy and fine-tuning prompt settings.

First, reducing OCR-induced noise is crucial. This can be achieved by utilizing advanced OCR algorithms, preprocessing PDF files to improve text clarity, and post-processing OCR outputs using natural language processing (NLP) techniques.

Second, fine-tuning the prompt settings can lead to more consistent and accurate metadata extraction. Experimenting with different temperature settings, prompt formats, and incorporating additional contextual information can optimize the extraction process.

By addressing these areas, the metadata extraction module will achieve more accurate, reliable, and comprehensive results.

7.2 Conclusion

In summary, this thesis presents the development of a comprehensive document management system that encompasses all essential features, including uploading, content extraction, version control, permission control, and document searching. The system streamlines the entire document handling process, from upload to retrieval, thereby significantly enhancing the user experience in managing documents and mitigating the challenges associated with traditional systems. Additionally, we constructed an ontology—a structured knowledge base—and utilized this ontology to refine user search queries, resulting in improved search outcomes.

References

[1] VNPT Vinaphone (n.d.). *Hệ thống quản lý văn bản và điều hành VNPT eOffice*. URL: https://www.truyensolieuvnpt.com/dich-vu-vnpt-vinaphone/he-thong-quan-ly-van-ban-va-dieu-hanh-vnpt-eoffice/132.html (visited on 12/10/2023).

[2] M-Files Corporation (n.d.[a]). *M-Files | Document Management Platform*. URL: https://www.m-files.com (visited on 12/10/2023).

[3] Tom Gruber (2008). "Encyclopedia of Database Systems". In: *Provides a definition of ontology as a technical term for computer science, tracing its historical context from philosophy and AI*.

[4] SNOMED International (n.d.). *SNOMED CT - Systematized Nomenclature of Medicine – Clinical Terms)*. URL: https://www.snomed.org/.

[5] Martin Hepp (n.d.). *GoodRelations*. URL: https://www.heppnetz.de/projects/goodrelations/.

[6] Severin Lemaignan et al. (2006). "MASON: A proposal for an ontology of manufacturing domain". In: *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*. IEEE.

[7] University of Edinburgh (2012). *Dynamic Ontology Repair*. URL: http://dream.inf.ed.ac.uk/projects/dor/.

[8] Robert Arp, Barry Smith, and Andrew D Spear (2015). *Building ontologies with basic formal ontology*. Mit Press.

[9] Ian Niles and Adam Pease (2001). "Towards a standard upper ontology". In: *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pp. 2–9.

[10] George A Miller (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

[11] Allan M Collins and M Ross Quillian (1972). “Experiments on semantic memory and language comprehension.” In.

[12] Christopher D Manning (2008). *An introduction to information retrieval*. Cambridge university press. Chap. 9.

[13] Elastic (2023). *What is semantic search?* URL: https://www.elastic.co/what-is/semantic-search (visited on 12/10/2023).

[14] Ontotext (2023). *What is a knowledge graph?* URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/ (visited on 12/10/2023).

[15] Django Software Foundation (2023). *Django documentation*. URL: https://docs.djangoproject.com/en/5.0/ (visited on 12/10/2023).

[16] Oracle Corporation (2023). *MySQL*. URL: https://docs.djangoproject.com/en/5.0/ (visited on 12/10/2023).

[17] Microsoft Corporation (n.d.[b]). *Microsoft SQL Server documentation*. URL: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16 (visited on 12/10/2023).

[18] PostgreSQL contributors (n.d.). *PostgreSQL documentation*. URL: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16 (visited on 12/10/2023).

[19] Ray Smith (2007). *Tesseract OCR Engine*. URL: https://tesseract-ocr.github.io/ (visited on 12/13/2023).

[20] PaddlePaddle (n.d.). *PaddleOCR*. URL: https://github.com/PaddlePaddle/PaddleOCR (visited on 12/13/2023).

[21] Amazon (n.d.). *Amazon S3 documentation*. URL: https://aws.amazon.com/documentation-overview/?nc2=h_ql_doc_do (visited on 12/15/2023).

[22] Elastic (n.d.). *Elasticsearch: Open Source Search & Analytics*. URL: https://www.elastic.co (visited on 12/15/2023).

[23] The Pivotal Software Foundation (n.d.). *RabbitMQ: Open Source Message Queuing*. URL: https://www.rabbitmq.com/ (visited on 12/15/2023).

[24] Redis Labs (n.d.). *Redis: Open Source NoSQL Database & Cache*. URL: https://redis.io/ (visited on 12/15/2023).

[25] Celery Project (n.d.). *Celery - Distributed Task Queue*. URL: https://docs.celeryproject.org/ (visited on 12/15/2023).

[26] Viet Trung Tran (n.d.). *pyvi: Python Vietnamese Core NLP Toolkit*. URL: https://github.com/trungtv/pyvi (visited on 12/15/2023).

[27] Kishore Papineni et al. (2002). "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

[28] Google Cloud (n.d.). *Evaluating models*. URL: https://cloud.google.com/translate/automl/docs/evaluate (visited on 05/15/2024).

[29] Nils Reimers and Iryna Gurevych (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084*.

[30] Ihechikara Vincent Abba (2022). *Crow's Foot Notation – Relationship Symbols And How to Read Diagrams*. URL: https://www.freecodecamp.org/news/crows-foot-notation-relationship-symbols-and-how-to-read-diagrams/ (visited on 12/15/2023).

[31] Quang Minh (2014). *Why is Vietnam's legal sysem "the most cumbersome and complicated in the world*. URL: https://baophapluat.vn/vi-sao-he-thong-phap-luat-nuoc-ta-cong-kenh-phuc-tap-nhat-the-gioi-post179873.html (visited on 12/15/2023).

[32] Ministry of Justice of Vietnam (n.d.). *Instructions for searching and using the Codification*. URL: https://phapdien.moj.gov.vn/qt/tintuc/Pages/huong-dan-su-dung-bo-phap-dien.aspx?ItemID=4 (visited on 12/15/2023).

[33] Van Phuc Vo (n.d.). *SimeCSEVietnamese: Simple Contrastive Learning of Sentence Embeddings with Vietnamese*. URL: https://github.com/vovanphuc/SimeCSE_Vietnamese (visited on 12/18/2023).

Appendix

Task assignment

Task	Responsible people
Authentication	Tran Le Trung Chanh
Building ontology	Nguyen Nam Kha
Querying ontology	Nguyen Nam Kha
OCR and extract metadata	Nguyen Ngoc Hoa
Document CRUD and version control	Tran Le Trung Chanh
Full-text search	Nguyen Ngoc Hoa
Semantic search	Nguyen Ngoc Hoa
Query expansion	Nguyen Ngoc Hoa
Audit log	Tran Le Trung Chanh
Permission control	Tran Le Trung Chanh
User interface	Nguyen Nam Kha
Deployment	All