HARAMAYA UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION SYSTEMS

Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University

BY:

NAME	ID
Mubarek Kemal	2065/13
Samuel Mamiru	2361/13
Samuel Kefelegn	2360/13
Dawit Tadele	3081/13

HARAMAYA UNIVERSITY

HARAMAYA, ETHIOPIA

HARAMAYA UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION SYSTEMS

Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University

A Project Submitted to the Department of Information Systems of Haramaya University in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Information Systems.

BY:

Name	**ID**
Mubarek Kemal	**2065/13**
Samuel Mamiru	**2361/13**
Samuel Kefelegn	**2360/13**
Dawit Tadele	**3081/13**

Advisor: Mr. Birhanu M.

Co-advisor: Mr. Kedir G.

December, 2023

HARMAYA UNIVERSITY, ETHIOPIA

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION SYSTEMS

CERTIFICATE OF APPROVAL

We here by certify that we have read and evaluated this Industrial Project I entitled Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University prepared under our guidance by the following students. We recommend that it be submitted as fulfilling the Industrial Project I requirement.

BY:

Name	**ID**
Mubarek Kemal	**2065/13**
Samuel Mamiru	**2361/13**
Samuel Kefelegn	**2360/13**
Dawit Tadele	**3081/13**

Name and Signature of members of Examining Board

Name	Title	Signature	Date
Birhanu Mekuriyaw	Advisor		
Kedir G.	Co- Advisor		
	Examiner 1		
	Examiner 2		
	Examiner 3		

DECLARATION

We therefore certify that all materials and data in this Proposal document are obtained and presented in compliance with guidelines of Information Systems Department. We also declare that; this work is entirely original and has never been presented or used by any other departments

Student Name:	**ID**	**Signature**
Mubarek Kemal	**2065/13**	______________
Samuel Mamiru	**2361/13**	______________
Samuel Kefelegn	**2360/13**	______________
Dawit Tadele	**3081/13**	_______________

This project has been submitted for examination with my approval as an advisor

Advisor Name: Mr. Birhanu M.	**Co- Advisor Name: Mr. Kedir G.**
Signature __________________	Signature _______________
Date ______________________	Date____________________

Acknowledgment

First and foremost, we would like to express our heartfelt gratitude to our lord the Almighty for guiding and enabling us to complete this work. We are immensely thankful to our advisor, Mr. Birhanu B., for their invaluable investment of time, providing us with guidance, insightful comments, and constructive suggestions throughout this document. We are also delighted to introduce our co-advisor, Kedir G., whose expertise and contributions have been instrumental in the development of this project. Their unwavering support, invaluable insights, and commitment have significantly contributed to our success. We extend our deepest appreciation to both Mr. Birhanu M. and Kedir G. for their continuous mentorship, guidance, and dedication. Additionally, we would like to express our gratitude to our friends, family, and colleagues for their unwavering encouragement and assistance throughout this journey.

Table of Contents

LIST OF FIGURES

LIST OF TABELS

LIST OF ABBRIVATION

BR-Business Rule

CRC- Class Responsibility Collaborator

DAO- Data Access Objects

ERD- Entity Relation Diagram

IDE - Integrated Development Environment

NLP- Natural Language Processing

ROI- Return on Investment

UI-User Interface

UML- Unified Modeling Language

1NF-First Normal Form

2NF-Second Normal Form

3NF-Third Normal Form

Abstract

The Web-Based Research and Project Documents Repository and Plagiarism Detector System is a comprehensive platform designed to meet the evolving needs of academic institutions. This system serves as a centralized repository for researchers, students, and Coordinator members, facilitating the seamless upload, storage, categorization, and retrieval of research documents. In addition to its document management capabilities, the system incorporates advanced plagiarism detection algorithms, comparing uploaded documents against existing databases and external sources to ensure academic integrity. Users receive timely notifications based on their preferences, alerting them to potential instances of plagiarism. With an intuitive and user-friendly interface, the system encourages collaboration and knowledge sharing within the academic community. The project adopts an iterative development approach, allowing for continuous enhancements to meet academic standards and user expectations. Ultimately, the Web-Based Research and Project Documents Repository and Plagiarism Detector System aims to foster a scholarly environment by providing a sophisticated yet accessible tool for document management and plagiarism prevention.

CHAPTER ONE

1. INTRODUCTION

This chapter presents a brief background of Project Documents Repository and Plagiarism Detector System and a statement of the problem that has been investigated. Additionally, it describes the objective of this project, and the significance of the project, and outlines the scope and delimitations of the project. To end, this chapter explains the structure of the project.

Due to the rapid advancement of the computer and network technologies, such as the Internet that enables anyone to access online contents anytime and from anywhere, academic integrity in the academic community is becoming a highly sensitive issue, especially among universities and research institutions. Plagiarism, on the other hand, is defined as a kind of academically dishonest behavior that will damage academic integrity Thus, it is needed to be resisted determinedly. However, plagiarism is not only an academic issue, but it extends to almost all industries. Occasionally, plagiarism occurs accidentally but most of the time it is the outcome of a conscious process[1]. The best definition of plagiarism might be that it is "the unacknowledged copying of documents or programs". To overcome the problem of plagiarism, large number researchers have worked on detecting plagiarism since the past decades through software detection methods. Plagiarism was originally detected manually (by hand) or by resembling previously consulted content. Today, the great number of the available online documents make it harder to detect plagiarism manually[2]. Therefore, there is an urgent need to produce automatic plagiarism detectors.

In an era dominated by information and technological advancements, academic institutions such as Haramaya University are faced with the challenge of efficiently managing and safeguarding the wealth of research and project documents produced by researchers and students. Recognizing the need for a robust and comprehensive solution, the Web-Based Research and Project Documents Repository and Plagiarism Detector System is introduced.

The Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University is designed to provide a comprehensive platform for researchers and students to manage, share, and analyze academic documents. This system aims to streamline document storage, retrieval, and plagiarism detection processes.

1.1. Background of the project

Haramaya University, a prominent institution in Ethiopia, is dedicated to advancing research and academic excellence. In the pursuit of these goals, the university recognizes the critical need for an efficient and technologically advanced system to manage research documents and uphold academic integrity. The proposed project aims to address this need by developing a comprehensive Web-Based Research and Project Documents Repository and Plagiarism Detector System tailored specifically for Haramaya University.

Currently, the university faces challenges in organizing, storing, and retrieving research documents efficiently. With an ever-growing volume of academic projects, papers, and theses, there is a pressing need for a centralized repository that streamlines document management, enabling easy access for researchers, students, and coordinator members. Maintaining academic integrity is paramount in higher education. The existing methods for detecting plagiarism may not fully meet the evolving needs of the university. The project addresses the necessity for a robust plagiarism detection system that goes beyond traditional methods, leveraging advanced algorithms to ensure the highest standards of academic honesty.

In the field of academic document management and plagiarism detection, there exists a wealth of research highlighting the importance of streamlined processes and document similarity plagiarism detection techniques. Studies have shown that institutions benefit significantly from systems that facilitate easy organization and access to research documents. Additionally, document similarities from the stored documents in the database plagiarism detection algorithms and technologies have showcased the potential for more accurate and comprehensive identification of instances of plagiarism. However, there is a gap in tailoring these technologies to the specific needs and context of Haramaya University.

The proposed system goes beyond basic document management and plagiarism detection. It serves as a collaborative platform that fosters knowledge sharing among researchers, students, and coordinator members. By providing an organized and accessible repository, the project aims to encourage collaboration and facilitate the exchange of ideas, ultimately enhancing the academic ecosystem at Haramaya University. The importance of academic integrity cannot be overstated. The project ensures that the university maintains its commitment to upholding the highest standards of honesty and originality in academic work. The document similarities plagiarism detection capabilities embedded in the system contribute to creating a culture of academic excellence and integrity. While existing systems and research provide valuable insights, the project recognizes the necessity of customizing technology to align with the unique requirements of Haramaya University. The system's development will be guided by an understanding of the university's academic landscape, ensuring that it seamlessly integrates into existing workflows and addresses specific challenges faced by the institution.

In conclusion, the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University is not merely a technological solution but a strategic initiative to enhance research capabilities, foster collaboration, and preserve the integrity of academic endeavors within the university community. This project aligns with the university's commitment to excellence and innovation in education and research.

1.2. Motivation of the project

The motivation behind the development of the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University stems from a multifaceted commitment to address existing challenges, bridging document management gaps, enabling efficient collaboration and knowledge sharing, upholding academic integrity, streamlining user experience, anticipating future technological needs, enhance the academic experience, and foster a culture of innovation and integrity within the university community.

1.3. Statement of the problem

Haramaya University, like many academic institutions, faces several challenges in the management and integrity of research and project documents. The existing document management practices at the university exhibit limitations that hinder the optimal utilization of academic resources. These challenges necessitate the development of the Research and Project Documents Repository and Plagiarism Detector System.

- **Inefficient Document Management**: The University relies on traditional methods of document storage, including physical archives and basic digital repositories. This approach leads to issues of disorganization, making it difficult for researchers and students to efficiently manage and retrieve academic documents. The lack of a centralized, user-friendly system impedes the seamless flow of information and collaboration.

- **Limited Accessibility and Knowledge Sharing**: Current document management practices hinder the accessibility of academic resources, limiting the potential for collaborative research and knowledge sharing. Researchers and students often face challenges in locating relevant documents, impeding interdisciplinary interactions and inhibiting the full utilization of the intellectual capital within the university.
- **Academic Integrity Concerns**: The University is grappling with the increasing threat of plagiarism, a challenge exacerbated by the interconnected nature of the academic landscape. The absence of a proactive mechanism to detect and prevent plagiarism poses a risk to the institution's reputation and compromises the ethical standards expected in scholarly work.
- **Technological Lag and User Experience**: The reliance on outdated systems contributes to a technological lag, hindering the university's ability to leverage modern solutions for document management. This, in turn, affects the overall user experience, making it cumbersome for researchers and students to upload, retrieve, and interact with academic documents seamlessly.
- **Lack of Future Preparedness**: The absence of a scalable and adaptable system leaves the university vulnerable to future technological demands. As academic and technological landscapes evolve, the current infrastructure may become obsolete, impeding the university's ability to stay at the forefront of academic innovation.

In light of these challenges, the Web-Based Research and Project Documents Repository and Plagiarism Detector System is proposed to address these issues comprehensively. By providing a technologically advanced, user-friendly platform, the system aims to enhance document management, foster collaboration, uphold academic integrity, improve user experience, and position Haramaya University for future advancements in academic technology.

1.4. Objectives of the project

1.4.1. General objective

To establish a centralized and user-friendly platform that enhances the efficiency of academic document management, facilitates seamless collaboration, upholds academic integrity through plagiarism detection, and anticipates future technological needs for Haramaya University.

1.4.2. Specific objectives

Implement Plagiarism Detection Model:

- ✓ Develop and integrate a machine learning-based plagiarism detection model into the system.
- ✓ Train the model using a diverse dataset of research documents to enhance accuracy.

Enhance Document Classification:

- ✓ Utilize machine learning algorithms to improve the automatic classification of documents into relevant categories.
- ✓ Train the classification model using labeled datasets representing various research topics.

Optimize Search Algorithm:

- ✓ Implement machine learning algorithms to enhance the efficiency and accuracy of the document search functionality.
- ✓ Incorporate natural language processing (NLP) techniques for semantic understanding during searches.

User Behavior Analysis:

- ✓ Apply machine learning techniques to analyze user behavior within the system.
- ✓ Identify patterns in document searches, downloads, and interactions to enhance user experience.

1.5. Methodology of the project

1.5.1. System Development Model: Waterfall model

For the development of the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University, the Waterfall model was chosen as the system development model. The Waterfall model is a linear sequential approach to software development that ensures each phase is completed before moving on to the next phase.

The following phases were followed during the development of the application:

1. Requirements Gathering and Analysis:
   - ✓ Identify the specific requirements for user registration, document upload, plagiarism detection, and user notification features.
2. System Design:
   - ✓ Design the overall system architecture, including the database schema, user interface layout, and the flow of information between system components.
3. Implementation:
   - ✓ Develop the user registration, document upload, plagiarism detection, and notification modules according to the design specifications.
4. Testing:
   - ✓ Perform unit testing for individual modules and integration testing to ensure that components work together seamlessly.
5. Deployment:
   - ✓ Deploy the Research and Project Documents Repository and Plagiarism Detector System for users at Haramaya University.
6. Maintenance:
   - ✓ Provide ongoing support, bug fixes, and updates to ensure the system remains functional and aligned with user needs.

By following the Waterfall model, the development of the Research and Project Documents Repository and Plagiarism Detector System for Haramaya University followed a systematic and well-defined process.

The model allowed for thorough requirements analysis, detailed design, and rigorous testing, resulting in a reliable and efficient system that met the project's objectives.

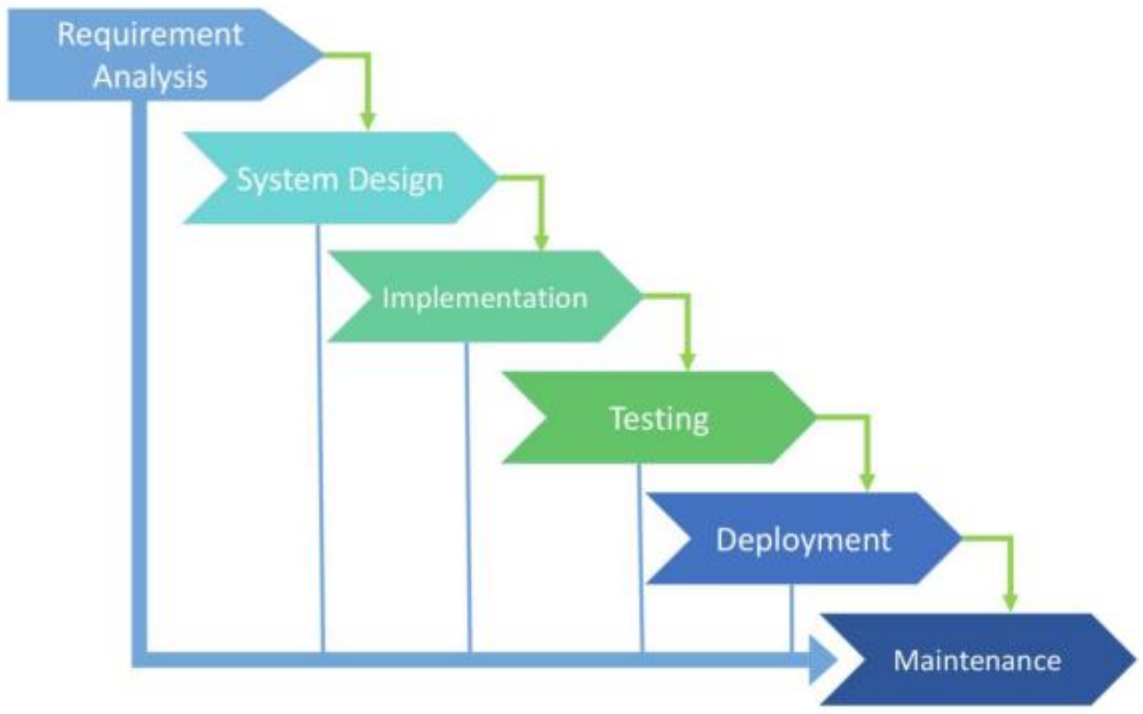


Figure 1 system development model

1.5.2. System development Tools:

1.5.2.1. Software Tools

The development of the Web-Based Research and Project Documents Repository and Plagiarism Detector System involves a set of tools to facilitate various aspects of the development lifecycle. Here's a list of essential tools for different stages of the project:

1. **Integrated Development Environment (IDE)**:
   - ✓ **Visual Studio Code:** is a versatile and lightweight IDE suitable for coding, debugging, and integrating with various programming languages.
2. **Database Management**:
   - ✓ IntegreSQL database, is well-suited for handling document-oriented data and efficiently storing research documents and metadata.
3. **Backend Framework**:
   - ✓ Django (Python): is a high-level Python web framework that provides a robust foundation for developing the backend of the system.

4. **Frontend Framework**:
   - ✓ **HTML, CSS and React.js:** is used for building user interfaces, providing a responsive and interactive frontend for the system.
5. **Software tools we used for documentation:**
   - ✓ Operating System: Windows
   - ✓ Microsoft word 2013: used for writing documentation sections
   - ✓ Web Server: Apache SERVER.
   - ✓ Browsers: used for running the code like Microsoft Internet Explorer, Google Chrome.
   - ✓ Draw.io: - used to design Sequence Diagram, Class Diagram Activity diagram and Use case Diagram of our system.
   - ✓ Figma: This application is used to develop and create UI prototypes. We used Figma to collaborate on the project's user interface prototype design.

1.5.2.2. Hardware Tools:

- Flash disk: For storing data what we have done
- Personal Computer: For performing every activity about the project
- Stationeries (pen, paper): for writing all necessary documentations associated with the project
- Note book: to take notes during data collection and for other documentations
- Printer: - for printing documentation

1.5.3. Data Collection Approaches

Data collection is a crucial aspect of any research, it is the process of gathering information or data from various sources in a systematic and structured manner. Data can be collected through various methods such as survey/questionaries, interviews, and observations. To gather the necessary data for the development of the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University, the following approaches will be utilized:

1. **Interviews:** Conduct one-on-one or group interviews with stakeholders, including researchers, students, Coordinator, and administrators, to understand their needs and expectations. We used this data collection method to gather information thoroughly and in detail, to exchange ideas and experiences, to extract data on a wide range of topics.

2. **Questionnaires:** Distribute questioners to a broader audience to collect quantitative and qualitative data on preferences, document management practices, and system requirements. We used this data collection method due to the usual large number of persons who respond to surveys, respondents can provide more frank and accurate responses and they provide similar definitions to all of the individuals who will be filling out the surveys. As a result, the data gathered may be measured with better precision.

1.6. Scope and Delimitation of the project

1.6.1. Scope

The Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University aims to address specific challenges related to academic document management and integrity within the university context. The scope of the project includes:

1. **User Registration and Authentication**: Development of a user-friendly registration system to allow researchers and students to create accounts securely.
2. **Document Upload and Management**: Implementation of a centralized repository for uploading, storing, and managing research papers, projects, and academic documents.
3. **Search and Classification Features**: Integration of advanced search and classification mechanisms to facilitate efficient document retrieval based on criteria such as title, author, keywords, and categories.
4. **Document Download and Access**: Provision of functionalities for users to download full documents or access abstracts, promoting easy and convenient document retrieval.
5. **Plagiarism Detection**: Incorporation of a robust plagiarism detection mechanism to calculate document similarity against existing documents.
6. **User Notification System**: Development of a notification system to alert users based on their preferences, especially when potential plagiarism is detected or new relevant documents are uploaded.
7. **User Interface Optimization**: Design and implementation of an intuitive and user-friendly interface for seamless navigation and enhanced user experience.
8. **Security and Privacy Measures**: Integration of secure user authentication and authorization mechanisms to ensure the privacy and integrity of user data and uploaded documents.

1.6.2. Delimitation

While the project aims to address the aforementioned aspects, certain limitations and delimitations are acknowledged:

1. **External Database Integration**: The system primarily focused on detecting plagiarism within the university's document repository. Integration with external databases cannot explored within the project scope.
2. **Language and Format Limitations**: The system primarily focused on English-language documents and specific file formats. Extensive support for multiple languages and varied document formats may be outside the immediate scope.
3. **Limited External Source Detection**: The plagiarism detection mechanism primarily compared documents within the system's database. Detection of plagiarism from external online sources have limitations due to the scope of the project.
4. **Hardware and Infrastructure Constraints**: The project assumes the availability of adequate hardware and infrastructure to host and deployed the system. Limitations in these areas may impact the system's performance.
5. **Policy and Legal Considerations**: Compliance with specific legal and policy frameworks related to data privacy and document sharing would be considered. However, the project may not delve deeply into the exhaustive exploration of all legal aspects.
6. **Integration with Existing University Systems**: While efforts would be made to align with existing university systems, the full integration with other university databases and platforms cannot be within the immediate project scope.
7. **Continuous System Maintenance**: The project would lay the foundation for ongoing system maintenance, but the continuous monitoring and update activities would be the responsibility of the university's IT support or designated personnel.

By acknowledging these scope and delimitation factors, the project aims to strike a balance between addressing immediate needs and laying the groundwork for potential future expansions and improvements within the university's academic environment.

1.7. Significance of the project

The Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University holds significant importance for the university, the researchers, the students, the administrators and the academic communities.

The implementation of the Web-Based Research and Project Documents Repository and Plagiarism Detector System holds immense significance for Haramaya University. The university can benefit from a centralized and streamlined approach to document management, which can contribute to the overall efficiency of academic processes. By integrating advanced plagiarism detection capabilities, the system will safeguard the institution's reputation by ensuring the highest standards of academic integrity. Additionally, the project aligns with the university's commitment to embracing modern technologies to enhance the overall academic experience for its community.

Researchers at Haramaya University can experience a transformative shift in their workflow and collaboration. The system can serve as a hub for storing and accessing research documents, providing a user-friendly interface for seamless navigation. Collaborative features can enable researchers to share knowledge, collaborate on projects, and stay informed about ongoing research initiatives. The plagiarism detection component cannot only protect the originality of their work but also foster a culture of academic honesty, enhancing the credibility of research produced by the university.

Students, as key stakeholders in the academic environment, will find the system instrumental in their educational journey. The repository will serve as a valuable resource for accessing academic materials, research papers, and relevant projects. This centralized access will facilitate a more organized and efficient approach to studying. The plagiarism detection system will educate students on the importance of academic honesty and provide a tool to validate the originality of their work, preparing them for ethical and responsible scholarly practices.

Academic administrators will find the system invaluable for managing and overseeing academic activities. The centralized repository ensures a standardized approach to document organization, making it easier for administrators to monitor research trends and allocate resources effectively. The plagiarism detection system provides an additional layer of quality assurance, helping administrators maintain the academic integrity of the institution and address any potential issues promptly.

The broader academic community, including Coordinator members, staff, and other stakeholders, will benefit from a more collaborative and transparent research environment. The system encourages interdisciplinary collaboration, breaking down silos and fostering a sense of community. As a result, knowledge exchange and innovation will thrive, contributing to the overall intellectual vibrancy of Haramaya University.

In summary, the Web-Based Research and Project Documents Repository and Plagiarism Detector System is not merely a technological upgrade; it is a catalyst for positive transformation across various facets of Haramaya University. The project's significance extends beyond streamlined processes, impacting the core values of academic integrity, collaboration, and innovation within the institution and its broader academic community.

1.8. Feasibility Assessment

The feasibility assessment of the Web-Based Research and Project Documents Repository and Plagiarism Detector System based on economic, technical, operational, and schedule considerations:

1.8.1. Economic Feasibility:

2. **Cost-Benefit Analysis:** Conduct a thorough cost-benefit analysis, considering development costs, maintenance expenses, and potential benefits such as increased academic integrity, collaboration, and streamlined document management.
1. **Return on Investment (ROI):** Assess the anticipated ROI by quantifying the expected benefits against the total project investment, ensuring that the project delivers value to stakeholders.
2. **Budget Constraints:** Evaluate the available budget and financial resources, ensuring that the project remains within budgetary constraints while delivering the intended functionalities.

1.8.2. Technical Feasibility:

1. **Technology Stack:** Assess the feasibility of the chosen technology stack (Django, React.js, etc.) by considering the team's expertise, community support, and alignment with project requirements.
2. **Plagiarism Detection Algorithm:** Evaluate the technical feasibility of integrating the chosen plagiarism detection API (Plagscan) into the system, ensuring its compatibility and effectiveness.
3. **Scalability:** Consider the technical scalability of the system to accommodate potential growth in user base and document volume, ensuring optimal performance.

1.8.3. Operational Feasibility:

1. **User Acceptance:** Assess the acceptance and enthusiasm of end-users (researchers, students, Coordinator) through surveys and feedback sessions to ensure that the system aligns with their needs and preferences.
2. **Training Requirements:** Identify the training needs of users to interact effectively with the system, ensuring a smooth transition and widespread adoption.
3. **Operational Impact:** Evaluate the potential impact on existing operational processes, identifying any adjustments or improvements needed for seamless integration.

1.9. Management Issue

1.9.1. Team configuration and management

The project development team is organized by three members. Each of the members has his/her own responsibility on the development system for timely and quality development. In case of workload there is high cooperation overcomes the workload.

No	**Name**	**Responsibility**
1.	Mubarek Kemal	All
2.	Dawit Tadele	All
3.	Samuel Mamiru	All
4.	Samuel Kefelegn	All

Table 1 Team configuration and management

1.9.2. Communication plan

1. The ejectives of communication are:
   - ✓ Ensure all team members and stakeholders are informed about project progress.
   - ✓ Facilitate transparent and timely communication.
   - ✓ Address issues promptly to avoid delays.
   - ✓ Foster collaboration and a shared understanding of project goals.
2. Communication channels:

- ✓ **Regular Team Meetings**: Weekly project team meetings to discuss progress, challenges, and upcoming tasks.
- ✓ **Project Management Tool**: Utilize a project management tool for task tracking, updates, and collaboration.
- ✓ **Documentation Repository:** Centralized repository on telegram for project documentation accessible to all team members.

3. Communication Schedule:
   - ✓ Team Meetings: Every Monday at 10:00 AM.
   - ✓ Project Management Tool: Continuous and real-time updates.

1.10. Organization of this work

The organization of this work is structured into three chapters to provide a comprehensive understanding of the project.

The first chapter serves as an introduction to the project and provides an extensive overview of the system. It includes background information about the organization and its history, setting the context for the project. The chapter also includes a statement of the problem, highlighting the specific challenges or shortcomings that the application aims to address. Clear objectives, both general and specific, are outlined to provide a clear direction for the project. Additionally, the chapter discusses the scope of the proposed application, detailing the boundaries and limitations of the project. The methodology used in the development process is also briefly explained. Lastly, the significance of the proposed application is emphasized, highlighting its potential impact and benefits.

Moving on to the second chapter, it focuses on user requirements and analysis. This chapter delves into the analysis of the currently operating existing system, identifying any limitations or bottlenecks that need to be overcome. It further explores the functional requirements, detailing the specific features and functionalities that the proposed application should encompass. Nonfunctional requirements, such as performance, security, and usability, are also addressed to ensure a holistic understanding of the project's requirements.

The third chapter provides a detailed overview of the system design for the proposed system. It outlines how the application intends to function, including the various components and modules involved. The chapter delves into the architecture and design decisions, explaining the rationale behind the chosen approach. This section provides a blueprint for the development process, guiding the implementation and ensuring a cohesive and well-designed system.

By organizing the document in this manner, readers will gain a comprehensive understanding of the project from its introduction, user requirements, and analysis to the detailed system design. This structure allows for a logical flow of information and facilitates effective communication of the project's objectives, scope, and methodologies.

CHAPTER TWO

2. USER REQUIRMENT AND ANALYSIS

2.1. Overview of the Existing Systems

The current state of document management and plagiarism detection at Haramaya University is characterized by decentralized and traditional methods. Research and project documents are stored across various platforms and physical locations, leading to challenges in organization and accessibility. Researchers and students encounter difficulties in navigating this fragmented system, hindering collaboration and efficient knowledge exchange. Additionally, the existing plagiarism detection mechanisms rely on manual checks and basic tools, making it challenging to ensure the highest standards of academic integrity.

In this decentralized environment, the university faces issues related to data redundancy, version control, and the potential for information silos. Researchers often struggle to locate relevant materials, impacting the efficiency of their work. Moreover, the limited plagiarism detection capabilities may not adequately address the evolving landscape of academic misconduct, necessitating a more advanced and automated approach.

Recognizing these shortcomings, the proposed system seeks to address the limitations of the existing infrastructure. By centralizing document management and incorporating advanced plagiarism detection algorithms, the project aims to streamline academic workflows, enhance collaboration, and reinforce the commitment to academic honesty. The transition from the current decentralized approach to a cohesive and technology-driven system is poised to bring about a paradigm shift in how Haramaya University manages and safeguards its academic knowledge assets.

2.2. Overview of Proposed System

The proposed Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University envisions a revolutionary advancement in the management of academic resources and the preservation of academic integrity. The system offers a comprehensive solution to the challenges posed by the existing decentralized infrastructure.

1. **Document Repository:**

The heart of the proposed system is a centralized repository that provides a user-friendly interface for researchers, students, and Coordinator to manage, organize, and access research and project documents. This repository will serve as a dynamic platform, facilitating efficient document storage, version control, and collaborative workflows. By consolidating academic resources in one accessible location, the system aims to streamline information retrieval and foster a collaborative environment among the university's academic community.

2. **Plagiarism Detection:**

The proposed system incorporates state-of-the-art plagiarism detection algorithms, going beyond traditional methods to ensure the highest standards of academic integrity. This advanced feature can automatically analyze and compare submitted documents against an extensive database, identifying potential instances of plagiarism. By leveraging cutting-edge technology, the system aims to enhance the accuracy and efficiency of plagiarism detection, providing a robust mechanism for safeguarding the originality of academic work.

3. **User-Friendly Interface and Collaboration Tools:**

To enhance user experience, the system will feature an intuitive and user-friendly interface, making it easy for researchers and students to navigate and interact with the platform. Collaborative tools will enable seamless knowledge sharing, project collaboration, and interdisciplinary interactions. The proposed system seeks to break down information silos, promoting a culture of openness and collaboration within Haramaya University.

4. **Academic Analytics and Reporting:**

The system will incorporate analytics and reporting tools to provide valuable insights into academic trends, research patterns, and document usage. These analytics will assist administrators in making informed decisions, allocating resources effectively, and monitoring the impact of academic initiatives. By offering a holistic view of academic activities, the system aims to contribute to strategic planning and continuous improvement.

In summary, the proposed system not only addresses the existing challenges faced by Haramaya University but also introduces a transformative approach to academic management. By centralizing document resources, implementing advanced plagiarism detection, and fostering collaboration, the system is poised to elevate the university's academic landscape, promoting efficiency, integrity, and innovation.

2.3. Functional requirement

Functional requirements outline the specific features and capabilities that the Web-Based Research and Project Documents Repository and Plagiarism Detector System must possess to meet the needs of Haramaya University. These requirements define the system's behavior and functionality, focusing on user interactions, document management, plagiarism detection, collaboration tools, and security measures. For instance, the user authentication and authorization requirement ensure secure access, while the document management requirement emphasizes efficient organization and version control. The collaboration features, notification system, and analytics components contribute to a user-centric and data-informed academic environment. Each functional requirement plays a crucial role in shaping the system's capabilities, contributing to a comprehensive solution tailored to the university's academic needs. The following is an overview of the functional requirements:

- ✓ Enable administrators to create user accounts for researchers, students, Coordinator members, and other relevant users.
- ✓ Enable administrators to remove user account information, including roles and permissions.
- ✓ Enable researchers, students, and Coordinator members to update their account information, such as contact details and research interests.
- ✓ Enable researchers and students to upload and download research papers and projects to the system.
- ✓ Enable the system to implement plagiarism detection algorithms for analyzing submitted documents.
- ✓ Design and implement a user-friendly interface with intuitive navigation for seamless interaction.

- ✓ Implement a robust search functionality, allowing users to search for documents based on various criteria.
- ✓ Develop a notification system to inform users of system updates, document changes, and plagiarism detection results.
- ✓ Enable access controls and security measures to protect sensitive academic materials.

2.4. Class Responsibility Collaborator (CRC)

Class Responsibility Collaborator (CRC) Class Responsibility Collaboration (CRC) cards are a brainstorming tool used in the design of Object-oriented software[3]. Its analysis to include the responsibilities and collaborations between the UI classes, Actor classes, and business classes in the Web-Based Research and Project Documents Repository and Plagiarism Detector System.

1. **UI Classes**

a. Class: Login

Responsibilities

- Handle the user's information and login into the system

Collaboration

- Actor (Researcher, Student, Coordinator and Administrator): Collaborates with different actor classes to present relevant information based on user roles.
- DocumentManager: Communicates to fetch and display document information.

b. Class: UIController

Responsibilities:

- Display the main dashboard for users, including Researchers, Students, Coordinator, and Administrators.
- Present document lists, metadata, and abstracts in a user-friendly manner.
- Enable users to upload research papers and projects.
- Implement an intuitive search interface for document retrieval.
- Capture user preferences and settings.
- Display notifications to users based on system activities.

Collaborators:

- Actor (Researcher, Student, Coordinator, and Administrator): Collaborates with different actor classes to present relevant information based on user roles.
- DocumentManager: Communicates to fetch and display document information.
- NotificationSystem: Displays notifications based on system activities.

c. **Class: Notification UI**

Responsibilities:

- Display real-time notifications to users.
- Provide options for users (Researchers, Students, Coordinator) to manage notification preferences.

Collaborators:

- Actor (Researcher, Student, and Coordinator): Collaborates with different actor classes to manage and display notifications.
- Notification System: Retrieves and displays notifications.
- User: Manages user-specific notification preferences.

2. **Business Class**

a. **Class: Document Manager**

Responsibilities:

- Retrieve document information from the backend.
- Manage document versions and revisions.
- Handle document uploads and storage.
- Coordinate with Plagiarism Detector for plagiarism checks.

Collaborators:

- UIController: Communicates to provide document information for display.
- Actor (Researcher, Student, Coordinator): Collaborates for document uploads and retrieval.
- Database: Stores and retrieves document-related data.
- PlagiarismDetector: Collaborates for plagiarism checks.

b. **Class: PlagiarismDetector**

Responsibilities:

- Compare documents for plagiarism.
- Generate plagiarism reports.
- Notify users of plagiarism alerts.

Collaborators:

- DocumentManager: Accesses document content for plagiarism comparison.
- UIController: Collaborates for notifying users about plagiarism alerts.
- Actor (Coordinator): Collaborates for utilizing plagiarism reports.
- User: Provides information about plagiarism incidents.

c. **Class: Database**

Responsibilities:

- Store document metadata, versions, and user information.
- Provide efficient data retrieval for search functionalities.

Collaborators:

- DocumentManager: Collaborates for storing and retrieving document-related data.
- SearchEngine: Collaborates for optimizing document retrieval.
- UIController: Stores and retrieves user-related data.
- Actor (Administrator): Collaborates for system administration.

d. **Class: SearchEngine**

Responsibilities:

- Implement advanced search functionalities.
- Optimize search algorithms for efficient document retrieval.

Collaborators:

- UIController: Collaborates for providing efficient search functionalities.
- DocumentManager: Optimizes document retrieval.

- Actor (Researcher, Student, and Coordinator): Collaborates for personalized search results.

d. Class: NotificationSystem

Responsibilities

- Manage user notification preferences
- Send real-time alerts for document uploads, plagiarism alerts, etc.

Collaborator

- UIController: Collaborates with the “UIController” class to display notifications in the user interface.
- User: Interacts with the “User” class to get and manage user-specific notification preferences.
- DocumentManager: Collaborates to trigger notifications related to document uploads and management.
- PlagiarismDetector: Collaborates to send alerts about plagiarism detection results.

3. Actor Class:

- **Responsibilities:** The Actor class represents the different types of users or system actors interacting with the Research and Project Documents Repository and Plagiarism Detector System. It manages user accounts and permissions, configure system settings, monitor system performance and security, upload academic projects and papers, search for educational resources, download relevant documents and utilize plagiarism detection reports for academic oversight.
- **Collaboration:** UI Class (to capture user interactions and display user-specific information), Business Class (to handle user requests and perform role-specific operations).

This extended CRC analysis emphasizes the collaborative relationships between UI classes, Actor classes, and business classes in the Web-Based Research and Project Documents Repository and Plagiarism Detector System. It provides a comprehensive view of how different components interact to fulfill their respective roles and responsibilities within the system.

CRC Diagram for User Interface Classes

Login<<UI>>	
Username Password Login ()	Login () Actor <<UI>> DocumentManager

Table 2 CRC diagram for login class

NotificationUI<<UI>>	
Display notification () Provide option ()	Display notification () Provide option () Actor <<UI>> NotificationSystem

Table 3 CRC diagram for NotificationUI class

UIController<<UI>>	
Display main dashboard and UI elements () Manage document presentations () Enable document uploads and searches () Capture user preferences and settings () Display notifications to users ()	Actor <<UI>> DocumentManager NotificationSystem

Table 4 CRC diagram for UIController

CRC Diagram for Business Classes

DocumentManager<<Business>>	
Retrieve document () Manage document () Handle document uploads and storage () Coordinate with Plagiarism Detector ()	UIController<<UI>> Actor <<UI>> Database PlagiarismDetector

Table 5 CRC diagram for Document Manager Class

PlagiarismDetector<<Business>>	
Compare documents () Generate report () Notify user ()	Actor <<UI>> UIController<<UI>> DocumentManager

Table 6 CRC diagram for PlagiarismDetector class

Database<<Business>>	
Store document () Data retrieval for search ()	Actor <<UI>> UIController<<UI>> DocumentManager SearchEngine

Table 7 CRC diagram for Database class

SearchEngine<<Business>>	
Implement advanced search () Optimize search algorithms ()	Actor <<UI>> UIController<<UI>> DocumentManager

Table 8 CRC diagram for SearchEngine class

NotificationSystem<<Business>>	
Manage user notification () Send real-time alerts ()	User <<UI>> UIController<<UI>> DocumentManager PlagiarismDetector

Table 9 CRC diagram for NotificationSystem class

CRC Diagram for Actor Classes

Researcher<<Actor>>	
Username Password Upload research papers and projects () Search document () Set preference ()	UIController<<UI>> DocumentManager NotificationSystem

Table 10 CRC diagram for Researcher class

Student<<Actor>>	
Username Password Upload academic projects and papers () Search document () Download document ()	UIController<<UI>> DocumentManager NotificationSystem

Table 11 CRC diagram for Student class

Coordinator<<Actor>>	
Username Password Upload academic projects and papers () Search document () Download document ()	UIController<<UI>> DocumentManager NotificationSystem

Table 12 CRC diagram for Coordinator class

Administrator<<Actor>>	
Username Password Manage user accounts and permissions () Configure system settings () Monitor system performance and security ()	UIController<<UI>> Database NotificationSystem

Table 13 CRC diagram for Administrator class

User<<Actor>>	
Username Password Authenticate and authorize () Set notification preferences () Interact with UI ()	UIController<<UI>> NotificationSystem

Table 14 CRC diagram for User class

2.4. Supplementary Specifications

2.4.1. Business Rules

Business rules are specific, actionable statements that define or constrain certain aspects of the operations, processes, or behavior within an organization. In the context of the Web-Based Research and Project Documents Repository and Plagiarism Detector System here are some potential business rules:

BR001. Document Upload:

- **Rule**: Only registered Researchers and Coordinator members are allowed to upload research papers and academic projects.
- **Rationale**: Ensures that only authorized individuals contribute content to maintain the quality and integrity of the repository.

BR002. Plagiarism Check:

- **Rule**: All uploaded documents must undergo a plagiarism check before being stored in the system.
- **Rationale**: Upholds academic integrity and ensures that only original content is available in the repository.

BR003. Access Control:

- **Rule**: Students can access academic resources, but they cannot upload academic publications.
- **Rationale**: Defines access permissions based on user roles, preventing unauthorized actions and maintaining a clear distinction between user types.

BR004. Notification Preferences:

- **Rule:** Users can customize their notification preferences to receive alerts for specific types of activities (e.g., document uploads, plagiarism alerts).
- **Rationale:** Enables users to tailor their experience and stay informed about activities relevant to their interests.

BR005. Document Versioning:

- **Rule**: Each document version must be stored, allowing users to track changes over time.
- **Rationale**: Facilitates document management and ensures that users can access and reference different versions as needed.

BR006. User Authentication:

- **Rule**: Users must authenticate themselves using secure login credentials.
- **Rationale**: Ensures the security and privacy of user accounts and their associated data.

BR007. System Monitoring:

- **Rule:** Administrators have the authority to monitor system performance, user activities, and security logs.
- **Rationale**: Supports system maintenance, troubleshooting, and security oversight.

BR008. Data Privacy:

- **Rule:** Personal information of users is kept confidential and is only accessible to authorized personnel.
- **Rationale**: Complies with privacy regulations and builds trust among users regarding the confidentiality of their data.

BR009. Plagiarism Report Availability:

- **Rule:** Coordinator members can access detailed plagiarism reports for documents uploaded by students.
- **Rationale:** Supports academic oversight and ensures that Coordinator members have the necessary information for evaluating academic work.

These business rules help define the operational and behavioral guidelines for the system, promoting consistency, security, and adherence to ethical and regulatory standards. Adjustments to these rules may be necessary based on the specific requirements and policies of Haramaya University.

2.5. Nonfunctional Requirements

Functional requirement describes what a software system should do, while Non-functional requirement is any requirement which specifies how the system performs a certain function. It will describe how a system should work and what limits there are on its functionality. Nonfunctional requirements generally specify the system's quality attributes or characteristics. These requirements play a critical role in ensuring that the system meets the expectations of its users and functions effectively[4]. Some of the non-functional requirement that needs to be met for the proposed system to operate correctly includes:

- **Performance**: The system should be able to handle concurrent document uploads by at least 100 users without significant performance degradation. The system should respond to user requests within an average latency of 2 seconds.
- **Scalability**: The system should be designed to scale horizontally to accommodate an increasing number of users and documents over time.
- **Reliability**: The system should have a minimum uptime of 99.9%.
- **Security**: User authentication and authorization mechanisms should follow industry best practices to prevent unauthorized access.
- **Usability**: The user interface should be intuitive and user-friendly, with an average user satisfaction rating of at least 80% in usability testing.
- **Maintainability**: The system should be designed with modularity, and updates should be deployable with minimal system downtime.
- **Compliance**: The system should comply with relevant data protection and privacy regulations (e.g., GDPR).
- **Interoperability**: The system should support integration with standard document formats (e.g., PDF, DOCX).
- **Response Time**: The system should respond to user requests within an average latency of 5 seconds.

2.5.1. Constraints

- **Technology Constraints**: The system must be developed using technologies and frameworks supported by the university's IT infrastructure.
- **Budgetary Constraints**: The project budget is limited to a specified amount.

- **Time Constraints**: The system must be deployed and operational within a specified timeframe.
- **Regulatory Compliance**: The system must comply with relevant data protection and privacy regulations, such as GDPR.
- **Infrastructure Constraints**: The system must operate within the hardware and network infrastructure provided by the university.
- **Security Policies**: The system must adhere to the university's security policies and standards.
- **User Authentication**: User authentication must integrate with the university's single sign-on (SSO) system.
- **Compliance with Academic Standards**: The plagiarism detection algorithm must align with recognized academic standards for assessing originality.
- **Browser Compatibility**: The system must be compatible with commonly used web browsers.
- **Documentation Requirements**: Comprehensive system documentation, including user manuals and technical documentation, must be provided.

These nonfunctional requirements and constraints collectively define the expectations, boundaries, and limitations for the Web-Based Research and Project Documents Repository and Plagiarism Detector System guiding its development and operation.

2.6. Use Case Modeling

Use case modeling is a technique used in software engineering to visually represent the interactions between a system and its external actors. It helps to identify, clarify, and organize system requirements from the perspective of users or external entities. It provides a foundation for understanding system functionality from a user perspective and helps in the identification of system requirements. The diagrams and descriptions are invaluable for communication between stakeholders and guiding the development process[5].

2.6.1. Essential Use Case Modeling

An essential use case is a high-level representation of the required behavior of a system, independent of any particular implementation or design decisions. It is not specific to an existing or proposed system but rather focuses on the fundamental functionality and goals of the system from a user's perspective. It describes the interactions between the actors and the system to achieve a specific goal or task. It is used to capture the essential requirements of the system and provides a basis for further development and refinement of the system's design and implementation[6]. Here are some essential use cases for the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University:

Actors: Researcher, Student, Coordinator and Administrator

Use Cases

1. **Upload Document:**
   - ✓ Actors: Researcher, Student, Coordinator
   - ✓ Description: Users can upload research papers and academic projects.
2. **Search Document:**
   - ✓ Actors: Researcher, Student, Coordinator
   - ✓ Description: Users can search for documents based on various criteria.
3. **Download Document:**
   - ✓ Actors: Researcher, Student, Coordinator
   - ✓ Description: Users can download documents or view abstracts.
4. **View Notifications:**
   - ✓ Actors: Researcher, Student, Coordinator
   - ✓ Description: Users can set and manage their notification preferences.
5. **Detect Plagiarism:**
   - ✓ Actors: Coordinator
   - ✓ Description: Coordinator can initiate plagiarism checks on uploaded documents.
6. **View Plagiarism Report:**
   - ✓ Actors: Coordinator, Administrator
   - ✓ Description: Coordinator, Students, Researchers and administrators can view plagiarism reports generated for documents.
7. **Manage User Accounts:**
   - ✓ Actors: Administrator
   - ✓ Description: Administrators can create, modify, and delete user accounts.

8. **Configure System Settings:**
   - ✓ Actors: Administrator
   - ✓ Description: Administrators can configure system settings, including security parameters.

Manage Security:

- ✓ Actors: Administrator
- ✓ Description: Administrator can maintain the system

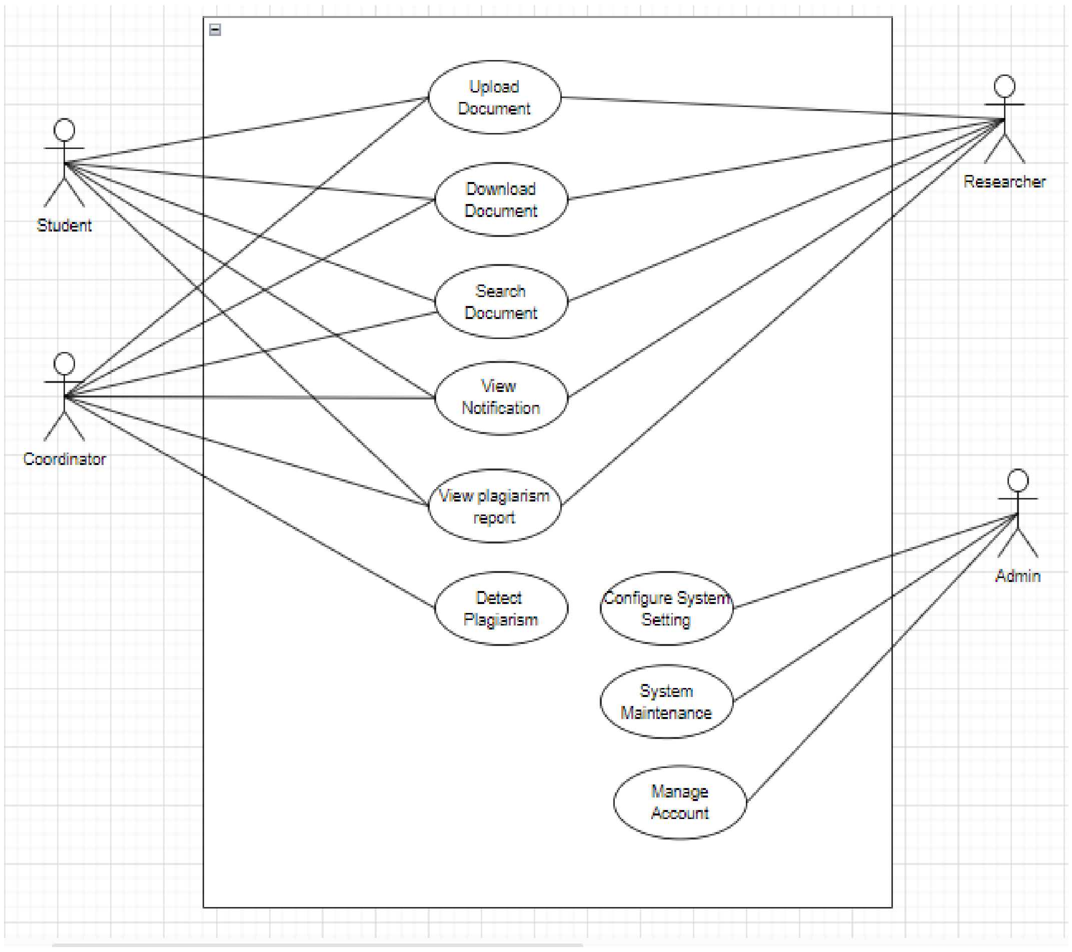


Figure 2 Essential Use Case Diagram

2.6.2. System Use Case Modeling

System use case modeling is also another technique used to model the behavior of the system like essential use case through the use of use case diagrams. System use case modeling involves identifying the actors and use cases that are required to support the system's functionality, and defining the relationships between them. But system use cases provide a detailed description of the functionality of the system, including the flow of events and the specific interactions between the actors and the system[7].

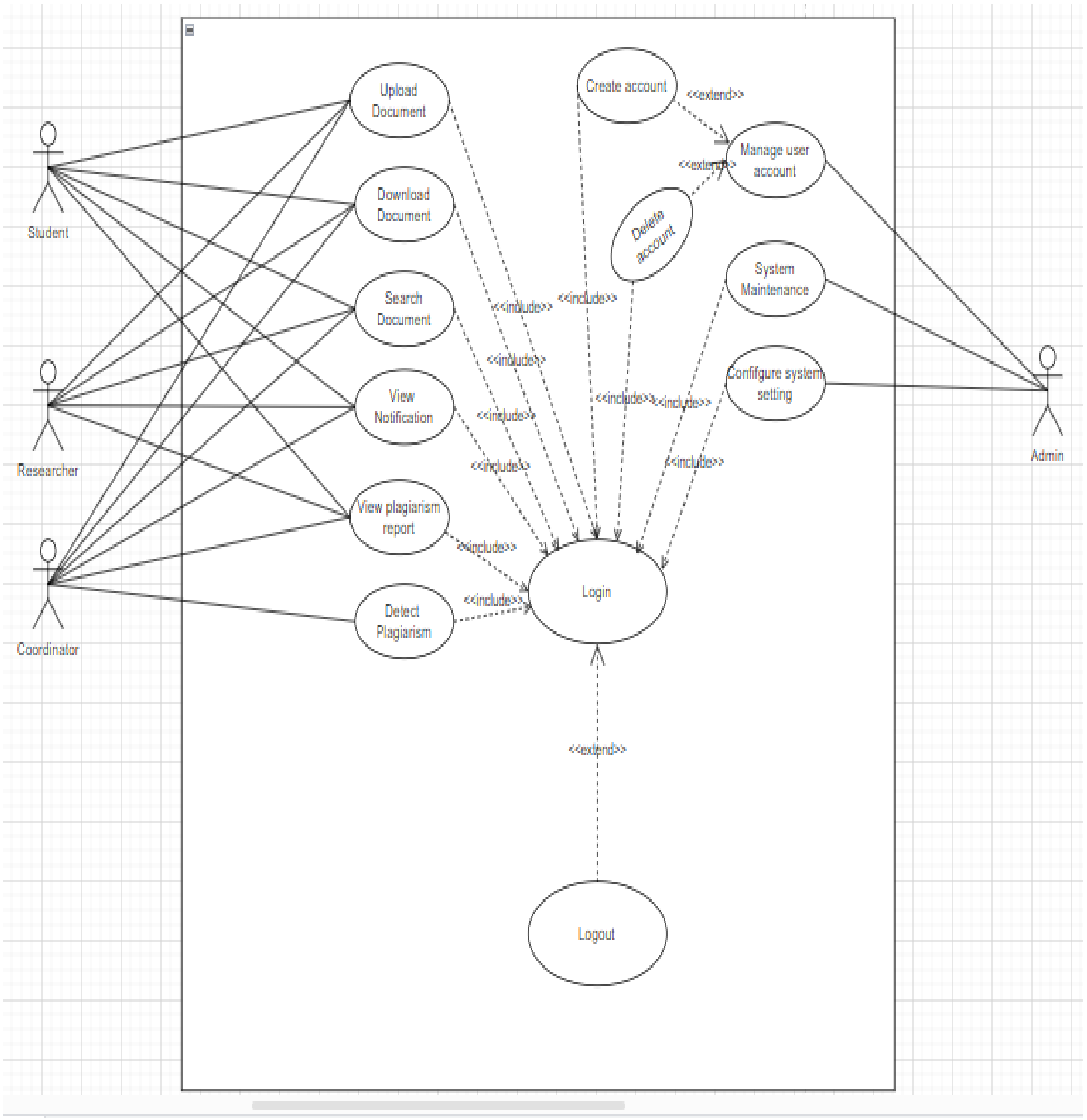


Figure 3 System Use Case Diagram

2.6.3. System Use Case Description

A system use case description is a document that outlines the functional requirements and behavior of a software system from the perspective of its users. A typical use case description includes the use case name and identification number, actors, preconditions, basic course of action, alternative flows, and post conditions:

Use Case Name	**ID**
Login	UC001
Upload document	UC002
Search document	UC003
Download document	UC004

Table 15 Use case name and ID

Use Case Name	Login
ID	UC001
Actor	Users (Researcher, Student, Coordinator)
Description	Users must login to access the system by entering their credentials for verification.
Precondition	The user must have a valid account for the system
Postcondition	User successfully logged in and gains access to the system features.
Basic course of action 1. User navigates to the login page. 2. User enters valid credentials (username and password). 3. System verifies the credentials. 4. If credentials are valid, the user is granted access.	
Alternative course of action A: Invalid Login Data 4. The System Displays error message for the user that he/ she entered invalid username or/ and password is invalid. 5. User use case resumes to step 1	
Alternative course of action B: if the user has no account	

4. The user reports to the administrator to have an account 5. The use case resumes to step 1

Table 16Use case description for Login

Use Case Name	Upload document
ID	UC002
Actor	Users (Researcher, Student, Coordinator)
Description	This use case allows users to upload their research papers and projects to the system for sharing and archiving. It involves providing document details such as title, author(s), abstract, keywords, and selecting the document type. The system then performs initial checks and stores the document in the repository.
Precondition	User must be logged in.
Postcondition	Document is successfully uploaded, and the system provides a confirmation
Basic course of action 1. User selects "Upload" from the dashboard. 2. User provides document details and uploads the file. 3. System validates and stores the document.	
Alternative course of action: If the user attempts to upload a document in an unsupported file format or type (e.g., non-PDF, non-Word), 3. The system detects the unsupported format. 4. The system displays an error message to the user indicating that the selected file format is not supported. 5. The system prompts the user to select a different file in a supported format.	

Table 17 Use case description for Upload document

Use Case Name	Search document
ID	UC003
Actor	Users (Researcher, Student, Coordinator)
Description	Users can search for relevant research documents within the repository based on specific criteria. The search includes entering

	keywords, filtering by category or subject, and exploring results. The system returns a list of matching documents for user review.
Precondition	User must be logged in.
Postcondition	User views a list of relevant documents based on the search criteria.
Basic course of action 1. User enters search criteria in the system. 2. System retrieves matching documents. 3. User reviews and selects documents for detailed view.	
Alternative course of action: If there is no match between the keyword and the stored document 3. The system displays a message indicating no documents match the search criteria. 4. The message may suggest refining the search terms, checking for typos, or trying alternative keywords.	

Table 18 Use case description for Search document

Use Case Name	Download document
ID	UC004
Actor	Users (Researcher, Student, Coordinator)
Description	This use case involves the user selecting a specific document and initiating the download process. The system ensures that the user has the necessary permissions to access and download the document.
Precondition	User is authenticated and has access to the document for download.
Postcondition	1. If permissions are valid, the user successfully downloads the document. 2. If permissions are invalid, the user is informed, and the download process is not completed.
Basic course of action 1. User navigates to the document details or search results. 2. User selects the document for download.	

3. System verifies user permissions for document access. 4. If permissions are valid, the system allows the user to download the document.
Alterative course of action: If the document has restricted access, 4. The system prompts the user with a message indicating limited permissions and provides options for further action.

Table 19 Use case description for Download document

5.4. User Interface Prototyping

Prototyping is an experimental process where design teams implement ideas into tangible forms from paper to digital. Teams build prototypes of varying degrees of fidelity to capture design concepts and test on users. With prototypes, you can refine and validate your designs so your brand can release the right products. The importance of this diagram is that to show the simple flow of application function. The boxes represent major user interface elements, modeled as you would instances/objects and the arrows represent the possible flow between them, modeled as you would transition inactivity diagrams[8].

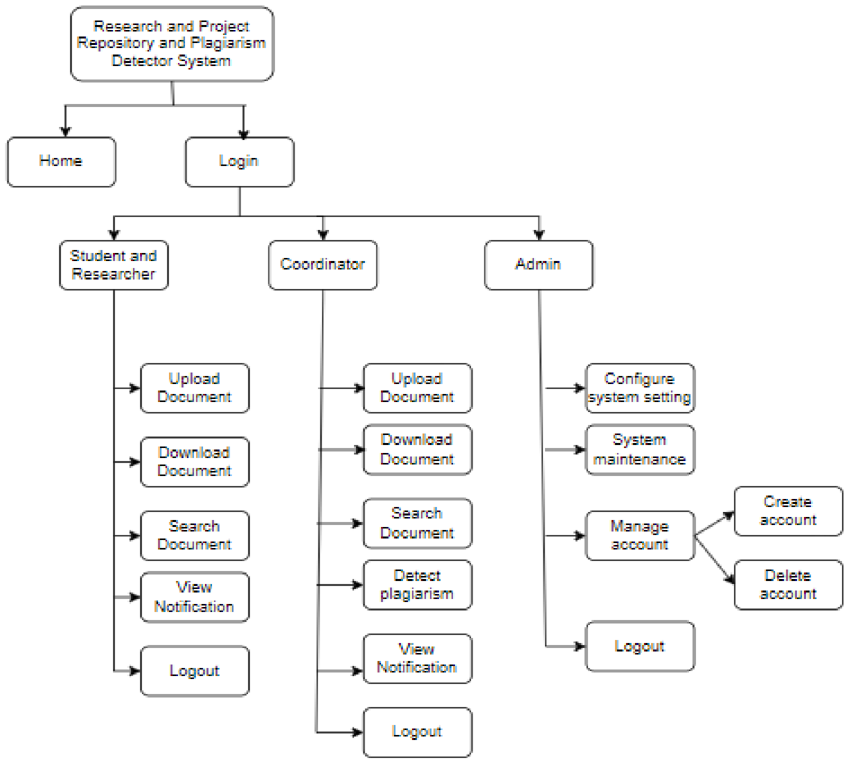


Figure 4 User Interface Prototyping

CHAPTER THREE

SYSTEM DESIGN

The System Design's goal is to provide sufficient and detailed information about the system to ensure that it is implemented consistently. From a functional standpoint, it is concerned with the system's overall view. This chapter includes Layering the Model, Class Modeling, User Interface Design, State Chart Diagram, ERD & Normalization of Table, Relational Persistent Modeling, Component Diagram and Deployment Diagram. So, mainly this phase concerned with the design part of our project to reduce the complexity of our system.

3.1. Class Type Architecture

Our project follows a layered architecture with different layers responsible for specific functionalities:

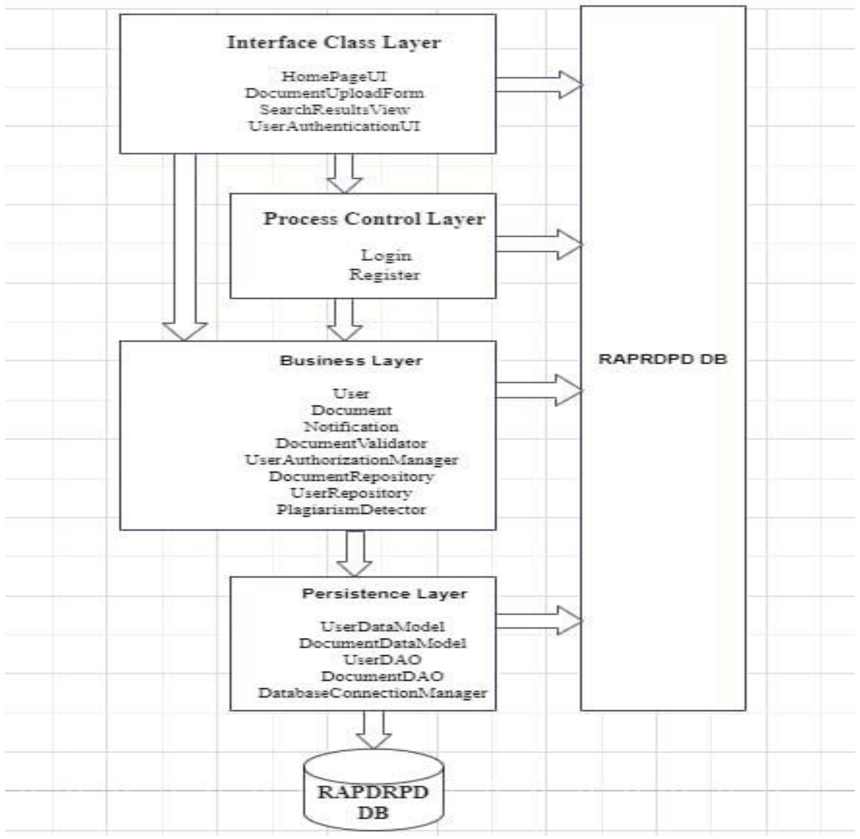


Figure 5 The Persistence Layer diagram

3.1.1. User Interface layer

The user interface layer represents the front end of the system user, and contains the actual GUI (graphical user interface) elements that users view and click. The user interface of the system has been easy to use by each user of the application with little training. This layer will be responsible for handling the presentation logic and user interaction. It will contain classes for building the UI components like screens, views, and widgets. It will also handle user input and output[9]. Our interface contains different buttons which guide the users what to click and fill to enter to their page to do what they want. So, our system user interface is easy to use. We also do our system clear which means the interface shows the user where to go in the system.

3.1.2. The Controller/Process Layer

The controller/process layer acts as an intermediary between the user-interface layer and the business/domain layer. It receives user input, processes requests, and communicates with the business layer to perform the necessary actions. It contains components such as controllers, APIs, and request handlers. Its main responsibility is to handle user requests and manage the flow of data and operations[10].

3.1.3. The Business/Domain Layer

The Business/Domain Layer contains the core business logic and rules of the system. It encapsulates the functionality and processes relevant to the domain of the application[11]. This layer is responsible for implementing the business rules, validations, and operations required for features such as User, Document, and Notification.

3.1.4. The Persistence Layer

The persistence layer is responsible for managing the storage and retrieval of data from the underlying database. It includes components such as data access objects (DAOs), repositories, and database connectors. This layer interacts with the database to store and retrieve information related to users, documents, etc. By following this layered architecture, the system separates the concerns and responsibilities of different components, making the system modular, maintainable, and scalable. Each layer has its specific role, promoting code reusability, and allowing for easier maintenance and future enhancements[12].

3.2. Class Modeling

A class diagram is a key element of the Unified Modeling Language (UML), which is used to describe the structure and behavior of a system. It is a type of static structure diagram that focuses on modeling the classes in the system, including their attributes, methods (or operations), and relationships with other objects in the system. The purpose of a class diagram is to provide a clear and visual representation of the system's structure, which can help developers understand how the system works and how its components relate to one another[13].

Inheritance Techniques: Inheritance is a technique in object-oriented programming that allows a subclass to inherit attributes and behaviors from a super class. There are two types of inheritance: single and multiple. Single inheritance involves one class inheriting from one other class, while multiple inheritances involve one class inheriting from multiple classes.

Association and Dependency Techniques: Association is a relationship between two or more classes where they are connected, but not dependent on each other. Dependency is a relationship between two or more classes where one class depends on another class to complete its functionality.

Aggregation and Composition Techniques: Aggregation is a relationship between two or more classes where one class is a part of another class, but can exist independently of it. Composition is a relationship between two or more classes where one class is composed of other classes, and cannot exist independently of it.

Modeling Methods during Design: When modeling methods during the design phase it’s important to consider the methods that will be used to interact with the class. Methods are functions that are associated with a class, and they define the behavior of the class.

Modeling Attributes During Design: Attributes are the data that are associated with a class. When modeling classes during the design phase, it's important to consider the attributes that will be associated with the class, as they define the state of the class.

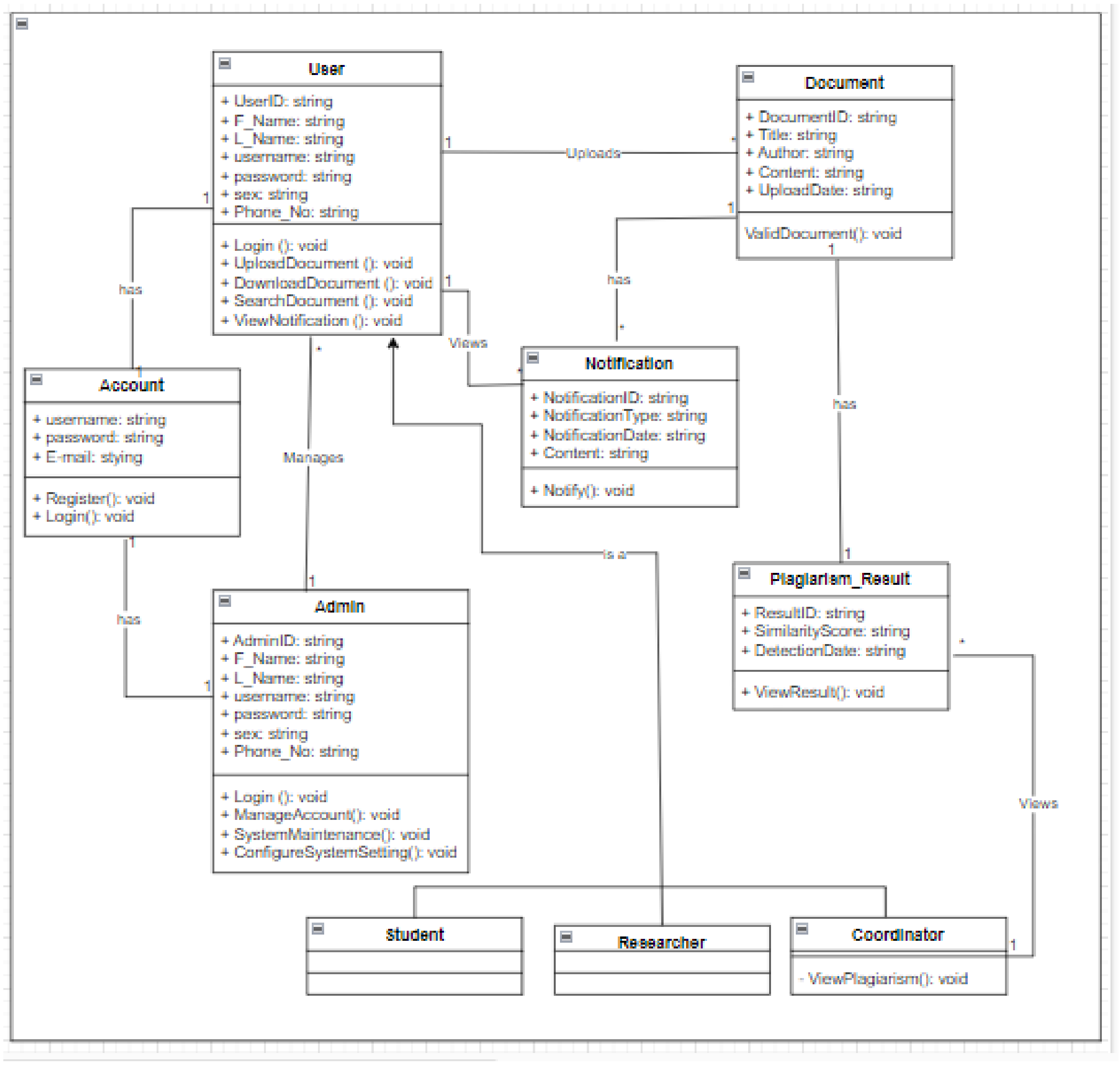
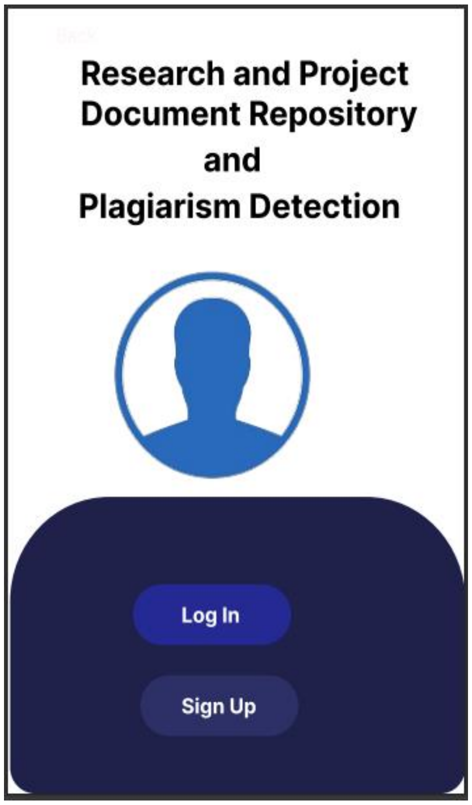


Figure 6 Class diagram

3.3. User Interface Design


Research and Project Document Repository and Plagiarism Detection
Log In
Sign Up

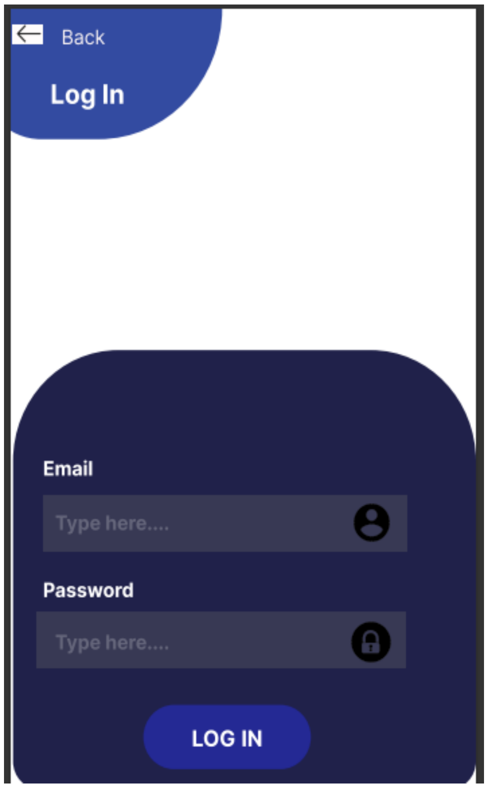


Figure 7 Home User Interface Design

Figure 8 Login User Interface Design

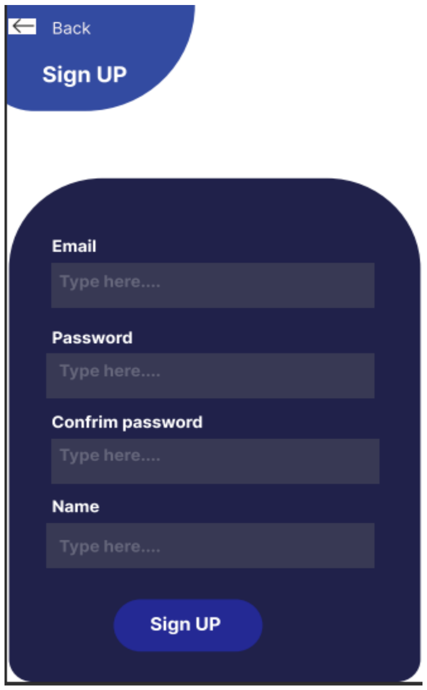


Figure 9 Signup User Interface Design

3.4. Sequence Diagrams

In UML (Unified Modeling Language), a sequence diagram is a type of interaction diagram that shows the interactions between objects or components in a system over time. It provides a visual representation of the sequence of messages that are exchanged between objects, and can be used to model the behavior of a system during specific use cases or scenarios. Sequence diagrams consist of objects or components represented by rectangular boxes, with arrows or lines indicating the messages or events that are passed between them. They are a powerful tool in software engineering for understanding and modeling the behavior of complex systems, and are widely used throughout the software development lifecycle[14].

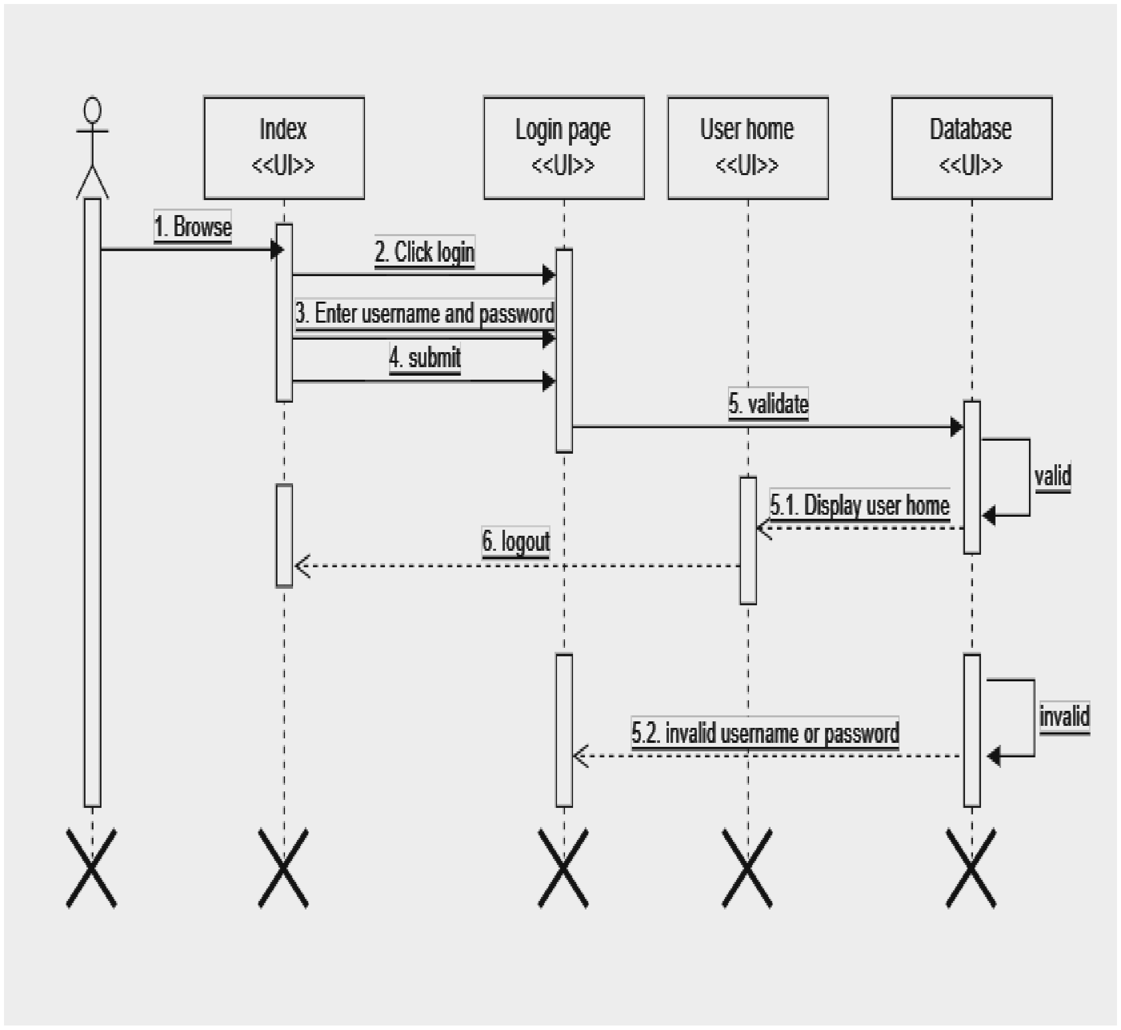


Figure 10 Login Sequence Diagram

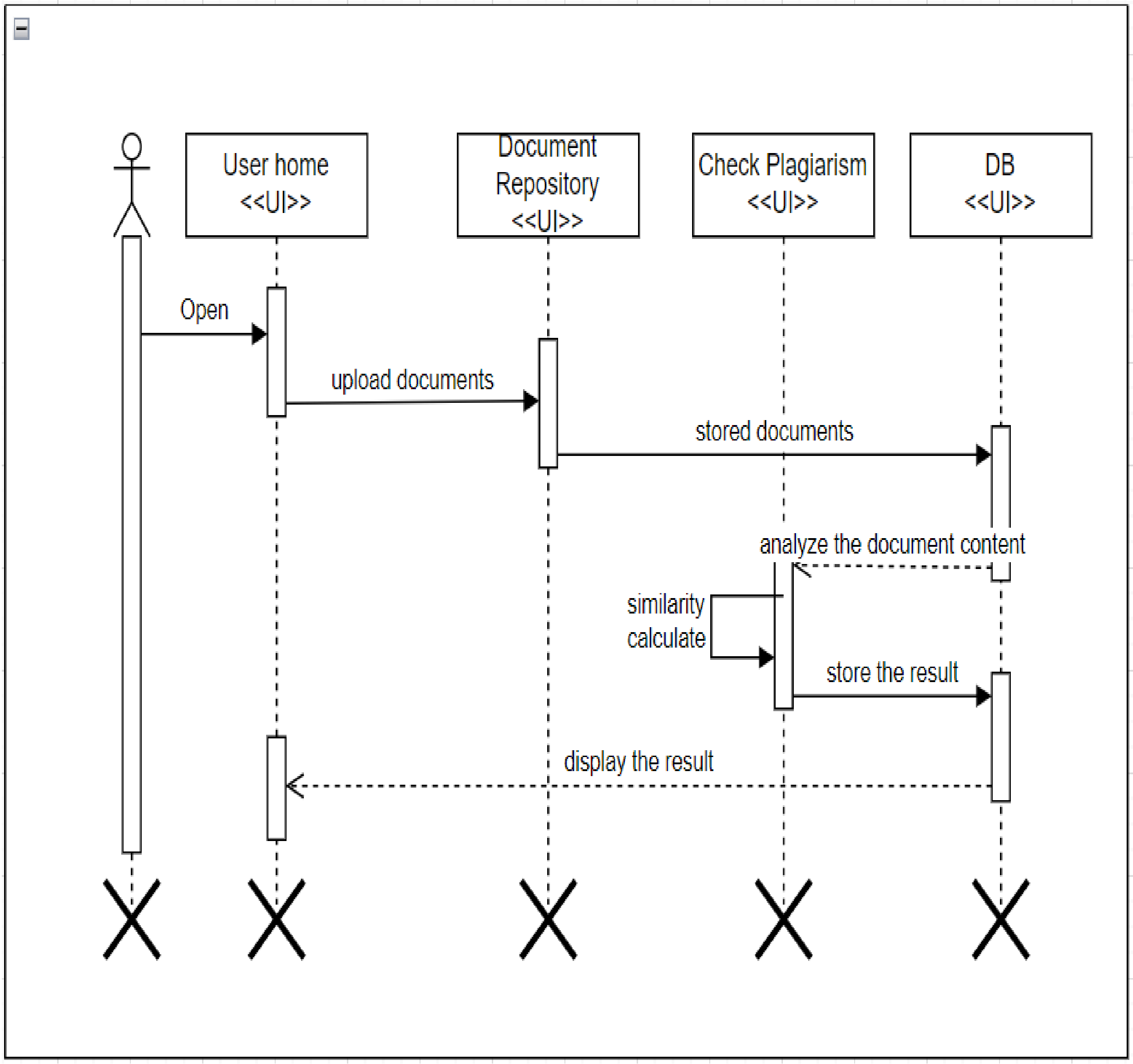


Figure 11 Document Upload and Plagiarism Detection Sequence Diagram

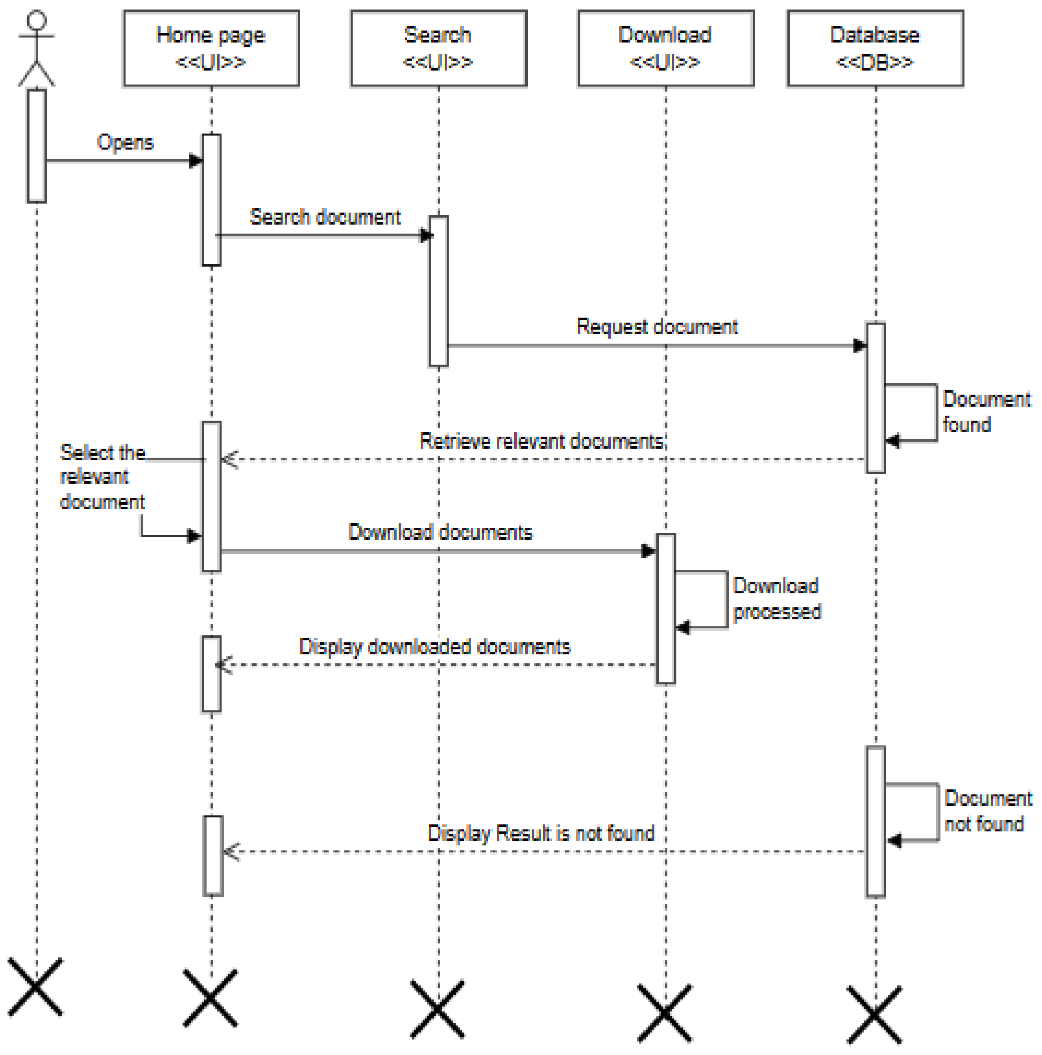


Figure 12 Search and Download Documents sequence diagram

3.5. Activity Diagrams

An activity diagram is a graphical representation of a system or process, often used to model the workflow and activities involved in a particular process. It provides a high-level view of the steps and an action involved in a process, and is often used to understand and analyze complex systems or business processes. Different symbols and notations are used to represent different elements of a process, such as actions, decisions, and transitions. Actions represent the steps or tasks involved in the process, while decisions represent the points where the process branches based on specific conditions. Transitions represent the flow of control between different activities[15].

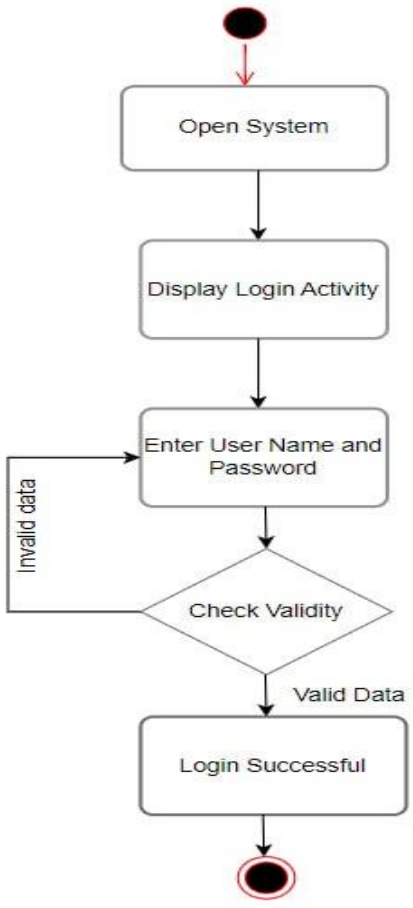


Figure 13 Login Activity Diagram

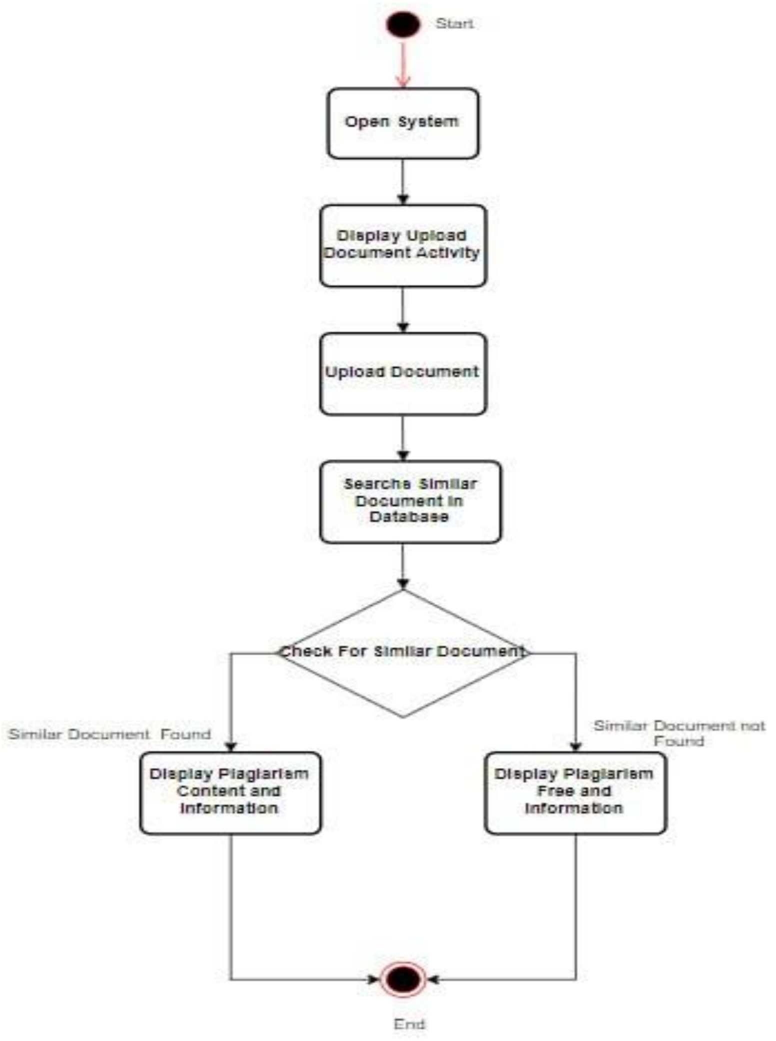

Start
Open System
Display Upload Document Activity
Upload Document
Searchs Similar Document in Database
Check For Similar Document
Similar Document Found
Similar Document not Found
Display Plagiarism Content and Information
Display Plagiarism Free and Information
End

Figure 14 Document Upload and Plagiarism Detection Activity Diagram

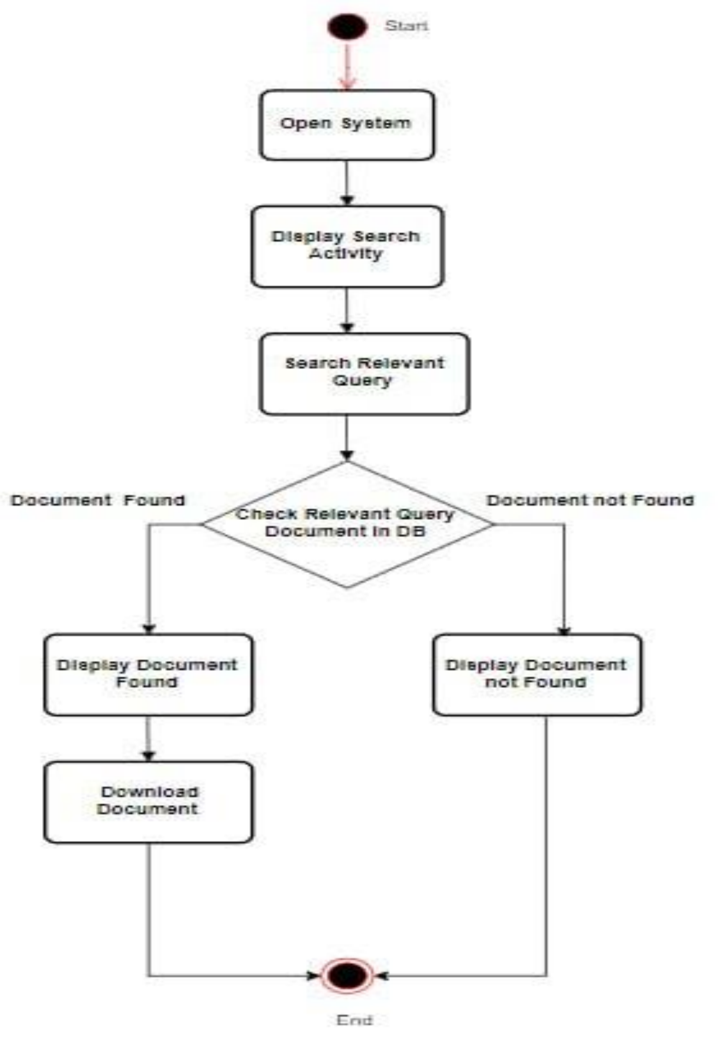


Figure 15 Search and Download Activity Diagram

3.6.State Chart Diagram

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So, the most important purpose of State chart diagram is to model life time of an object from creation to termination. State chart diagram is used to describe the states of different objects in its life cycle. So, the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. Since, the system has its own states while the user operates on it[16].

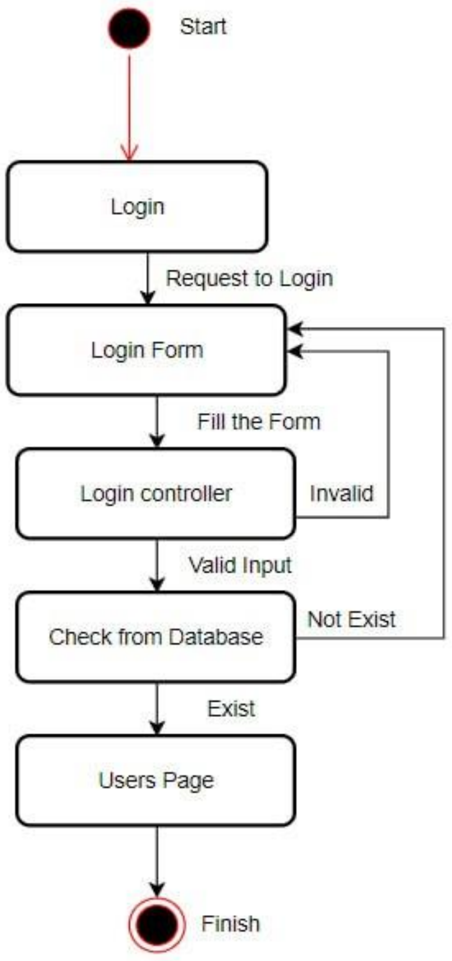


Figure 16 : Login State chart diagram

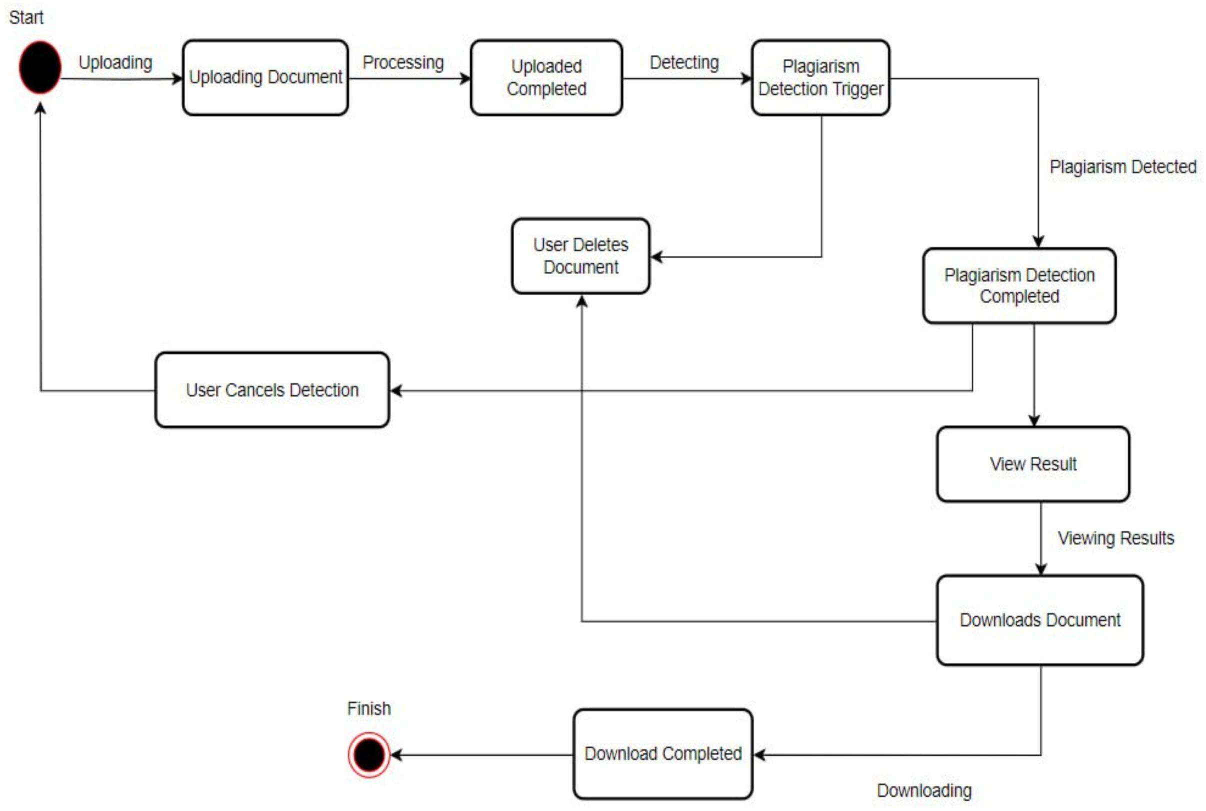


Figure 17 Document State Chart Diagram

3.7.ERD and Normalization of Table

3.7.1. Entity Relation Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes[17].

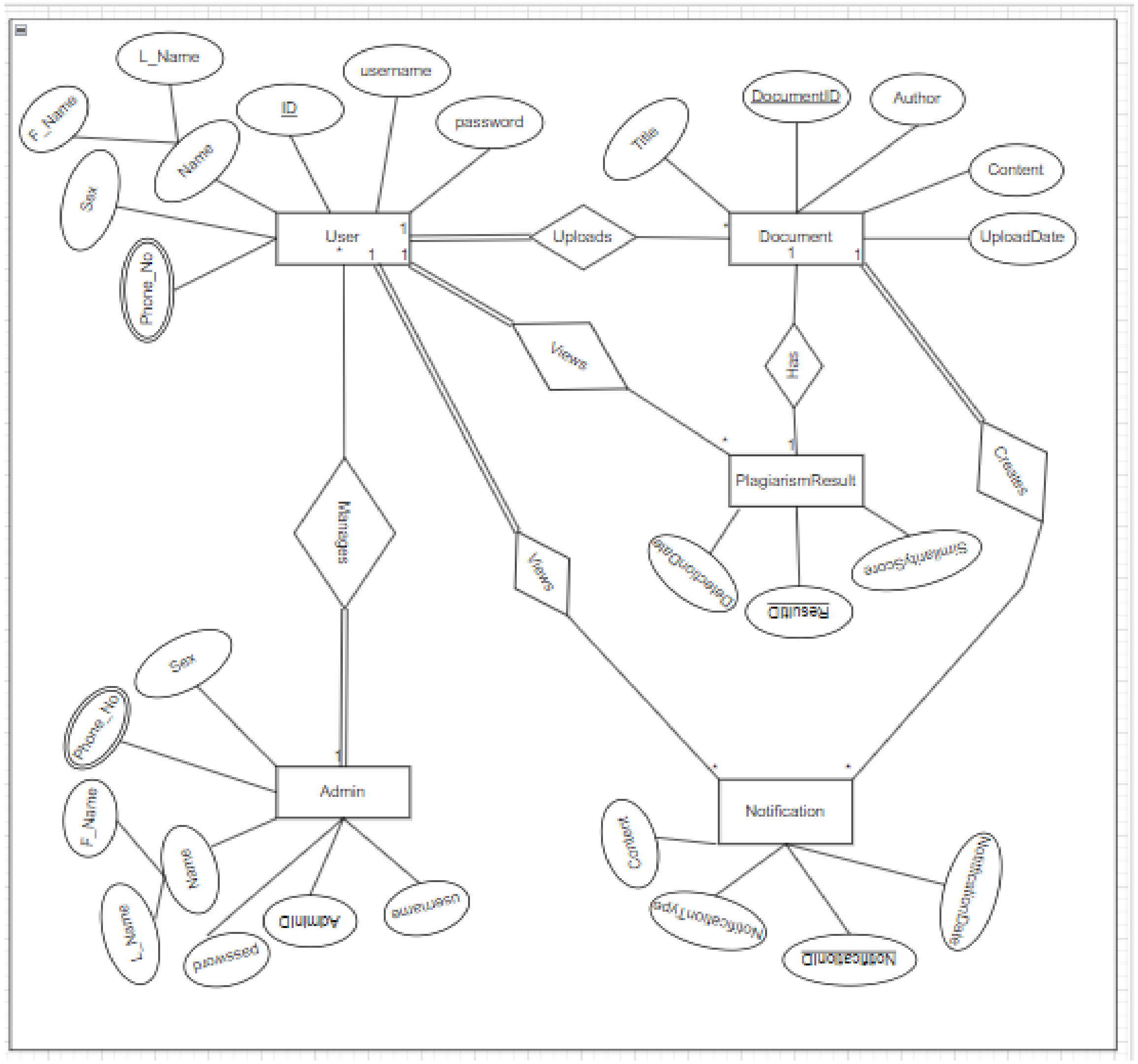


Figure 18 ERD diagram

3.7.2. Normalization of Tables

Normalization of tables is a process in database design that aims to minimize data redundancy and improve data integrity by organizing data into multiple tables and eliminating data dependencies. It involves applying a set of rules called normal forms to ensure that each table represents a single logical entity and that the data is stored in the most efficient and logical manner. The normalization process typically involves the following normal forms:

1. **First Normal Form (1NF):** Ensures that each column in a table contains only atomic values (indivisible values) and there are no repeating groups of columns.

2. **Second Normal Form (2NF):** Builds upon 1NF and eliminates partial dependencies by ensuring that each non-key column is fully dependent on the primary key.
3. **Third Normal Form (3NF):** Builds upon 2NF and eliminates transitive dependencies by ensuring that each non-key column is only dependent on the primary key and not on other non-key columns.

Normalization helps improve data integrity by reducing data redundancy and inconsistencies. It also allows for efficient data storage and retrieval, as related data is stored in separate tables and can be accessed through well-defined relationships. In the context of our system, normalization would involve analyzing the entities and their attributes, identifying functional dependencies, and designing the tables in a normalized form to ensure data integrity and efficiency in storing and retrieving data[18].

User						
User_ID	F_Name	L_Name	username	password	sex	Phone_No

Table 20 User Normalized Table

Admin							
Admin_ID	F_Name	L_Name	username	password	sex	Phone_No	User_ID

Table 21 Admin Normalized Table

Document					
Document_ID	Title	Author	Content	Upload_Date	User_ID*

Table 22 Document Normalized Table

Plagiarism Result			
Result_ID	Similarity_Score	Detection_Date	Document_ID*

Table 23 Plagiarism Result Normalization Table

Notification					
Notification_ID	Notification_Type	Notification_Date	Content	User_ID	Document_ID

Table 24 Notification Normalized Table

3.8.Relational Persistent Modeling

Persistent modeling helps ensure that the data is organized, stored, and retrieved efficiently, and that it accurately represents the entities and their relationships in the system. It involves making design decisions regarding table structures, data types, primary keys, foreign keys, constraints, and other aspects of database design. By following good practices in persistent modeling, our system can achieve data integrity, optimize database performance, and provide a solid foundation for data storage and retrieval operations[19].

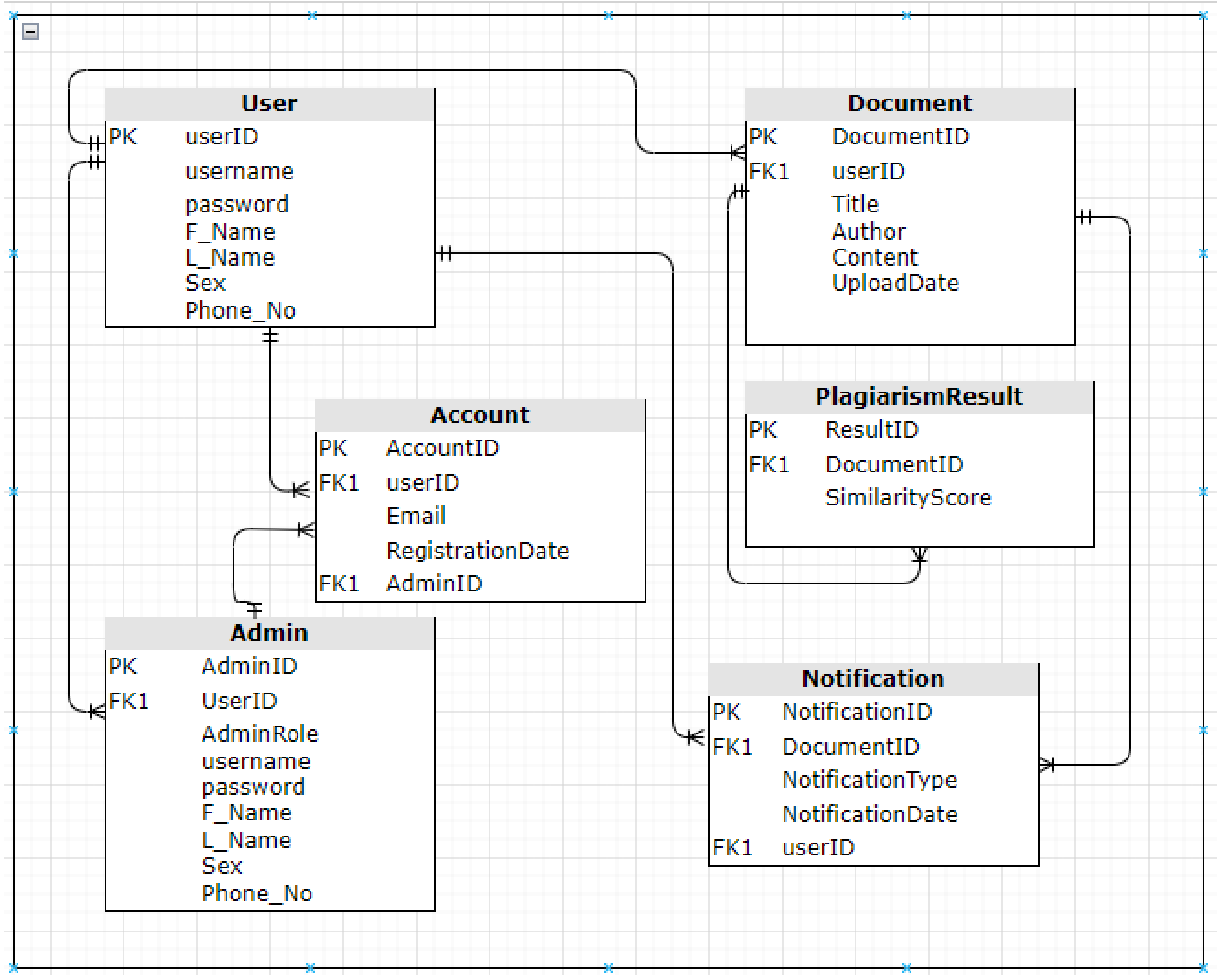


Figure 19 Relational Persistent Diagram

3.9.Component Diagram

A component diagram is a type of UML (Unified Modeling Language) diagram that depicts the structure of a system or software application in terms of its components and their relationships. It is used to illustrate the organization and dependencies among the various parts of a system. In a component diagram, each component is represented as a rectangle with the component name written inside. The components can be connected by lines that represent the interfaces between them. The interfaces can be either provided or required and may be associated with different types of relationships, such as composition, aggregation, or dependency[20].

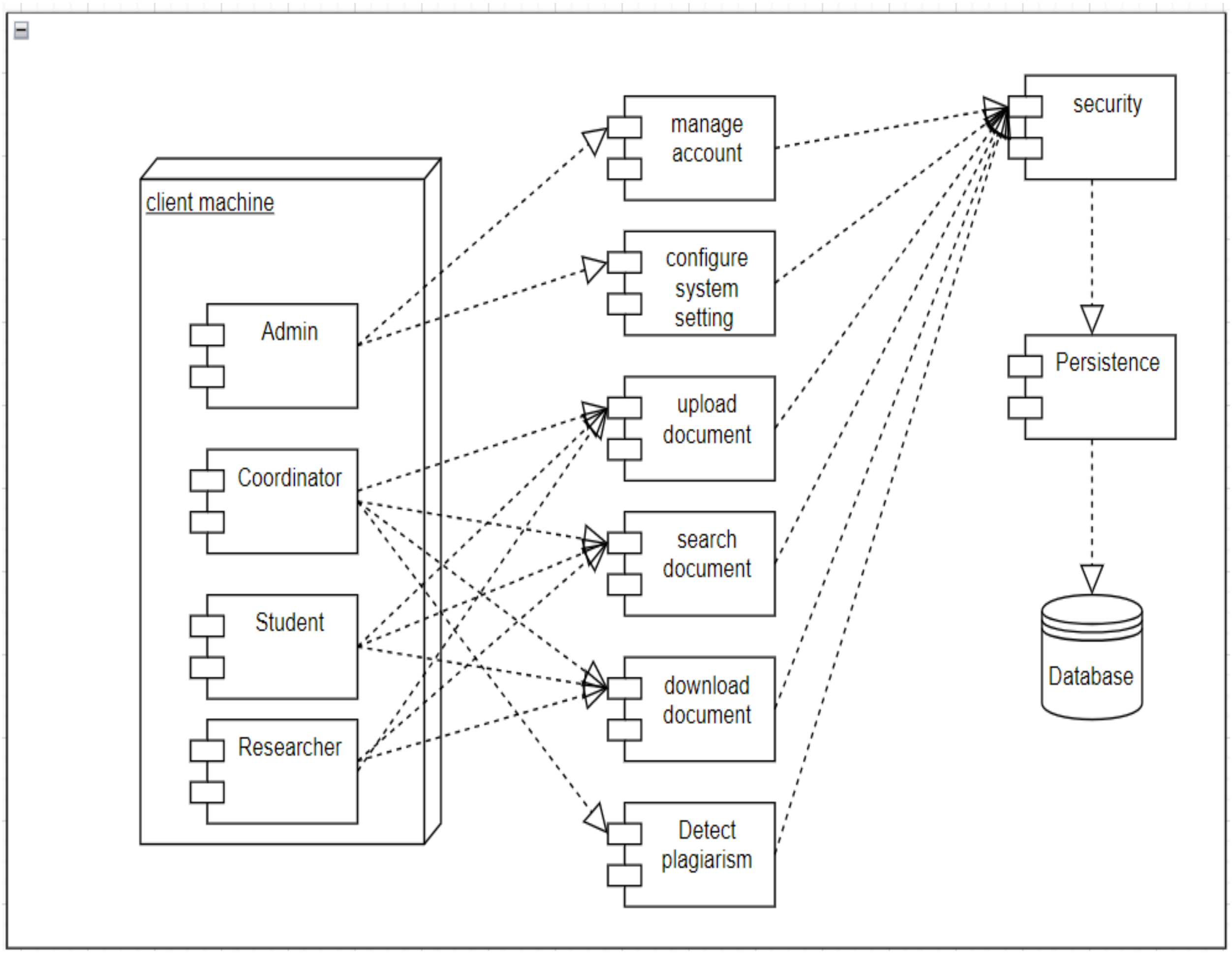


Figure 20 Component Diagram

3.10. Deployment Diagram

A deployment diagram is a type of UML diagram that shows the physical architecture of a system. It depicts the hardware and software components of a system and how they are connected and deployed across different nodes. In a deployment diagram, nodes represent the physical components of a system, such as servers, computers, or devices. Each node can have multiple components that are deployed on it, and these components can be either hardware or software. The connections between nodes are represented by communication paths, which show how different components interact with each other[21].

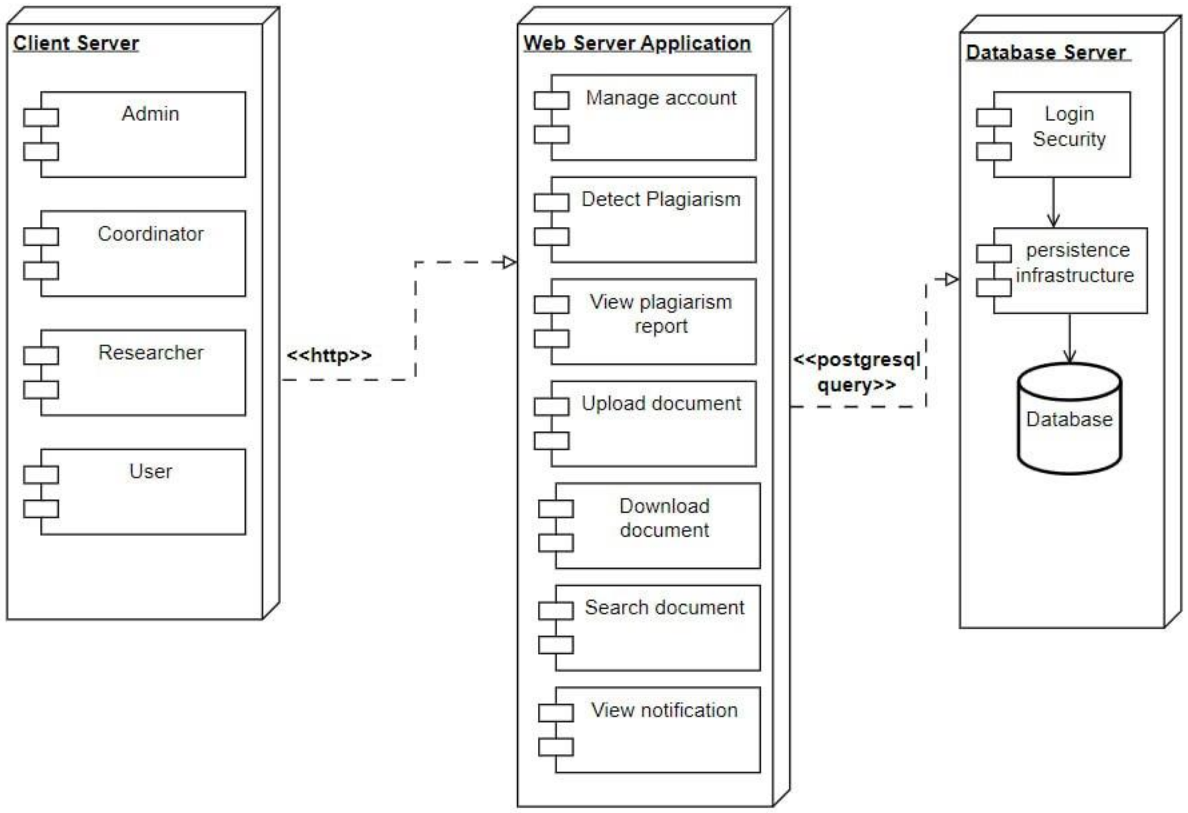


Figure 21 Deployment Diagram

CHAPTER FOUR

4. IMPELEMENTATION

4.1. Overview

This chapter delves into the practical aspects of developing the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University. It outlines the development tools employed, provides sample code snippets, and details the testing strategies implemented to ensure system quality and user satisfaction. It covers system development tools, sample code of the system/prototype, testing procedures, client feedback, user manuals, and concludes with recommendations.

4.2. System Development Tools

Frontend Development:

- **HTML:** Defines the structure and content of your web pages, acting as the foundation for everything you see on the screen (text, images, forms, etc.).
- **CSS:** Controls the visual appearance of your web pages, allowing you to style elements like fonts, colors, layouts, and more. It's like adding a coat of paint and styling your HTML content.
- **Bootstrap:** Provides a set of pre-built styles and components for web development, making it faster and easier to create responsive designs that adapt to different screen sizes (phones, tablets, desktops). Think of it as a toolbox with ready-made components to save you time and effort.
- **JavaScript:** Adds interactivity and dynamic behavior to your web pages. It allows you to create features like user input validation, animations, and real-time updates without needing to refresh the entire page. It's like adding special effects and functionality to your web pages.

Backend Development:

- **Python:** A general-purpose programming language known for its readability and ease of use. It's a powerful tool for building web applications and interacting with databases.

- **Django:** A high-level Python web framework that simplifies the development process by providing a pre-built structure and handling many common web development tasks. It's like using pre-made building blocks to construct your web application faster and more efficiently.
- **PostgreSQL:** A robust open-source relational database management system (RDBMS) used to store and manage the data associated with your system. It allows you to organize and access information like uploaded documents, plagiarism scores, and user accounts. It's like the filing cabinet for your application's data.

In summary, we used these tools because they work well together to create a web-based system:

- The frontend tools (HTML, CSS, Bootstrap, and JavaScript) build the user interface that users interact with.
- The backend tools (Python, Django) handle the logic and functionality behind the scenes, interacting with the database (PostgreSQL) to store and retrieve data.

Common Libraries:

- django.shortcuts: Offers convenience functions for rendering templates, handling redirects, and more.
- .models: Provides a layer for interacting with the database using Django's model classes.
- os: Enables interaction with the operating system for file system operations.
- django.http: Facilitates sending HTTP responses from Django views.

Plagiarism Detection Libraries:

- docx (Python-docx): Used for parsing and working with Microsoft Word (DOCX) documents.
- PyPDF2: Employed for extracting text content from PDF files.
- sklearn.feature_extraction.text.CountVectorizer: Creates a numerical representation of text data based on word frequencies.
- sklearn.metrics.pairwise.cosine_similarity: Calculates cosine similarity between text documents to determine plagiarism likelihood.

4.3. Sample Code Snippets

1. Detector app (views.py):

```
from django.shortcuts import render, redirect
from .models import FileRepository
import os
from django.http import HttpResponse
from docx import Document  # Optimized import for readability
import PyPDF2
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from django.core.exceptions import ValidationError
from django.db.models import Q
from django.shortcuts import get_object_or_404
def home(request):
       return render(request, 'index.html')
def about(request):
       return render(request, 'about.html')
def coordinator(request):
       return render(request, 'coordinator.html')
def download_file(request, file_id):
    file_repository = get_object_or_404(FileRepository, id=file_id)
    file_path = file_repository.file.path
    with open(file_path, 'rb') as file:
        response = HttpResponse(file.read(), content_type='application/octet-stream')
        response['Content-Disposition'] = 'attachment; filename=' + file_repository.file.name
        return response
def user(request):
     if request.method == 'GET':
        search_query = request.GET.get('search_term')
        if search_query:
            # Perform case-insensitive search using icontains
            search_results = FileRepository.objects.filter(
                Q(title__icontains=search_query) | Q(file__icontains=search_query)  # Search both title
and filename
            )
        else:
            search_results = []  # Empty list if no search query
        print(search_results)
        return render(request, 'student_and_reeasercher.html', {'search_results': search_results})
     return render(request, 'student_and_reeasercher.html')
def contact(request):
       return render(request, 'contact.html')
def upload_file(request):
    if request.method == 'POST':
        category = request.POST.get('Category')
        uploaded_file = request.FILES.get('file_name')
        if  not uploaded_file:
            return HttpResponseBadRequest('Please select a category and upload a file.')
        try:
            # Check for uniqueness based on file name (recommended)
            # existing_file = FileRepository.objects.filter(file__name=uploaded_file.name).first()
            # if existing_file:
            #      return HttpResponseBadRequest('A file with the same name already exists. Please
rename the file and try again.')
            # Save the file and create a new record if unique
            # 333443345676543
            new_file = FileRepository(category=category,file=uploaded_file)
            new_file.save()
            print('hi')
            return render(request,'file_upload.html')  # Replace with your success URL
        except ValidationError as e:
            error_message = str(e)  # Extract validation errors for better user feedback
            return render(request, 'upload_form.html', {'error_message': error_message})
    return render(request, 'file_upload.html')
def upload_file_for_plagiarism_check(request):
    if request.method == 'POST':
        file = request.FILES['file']
        # Preprocess the uploaded file
        uploaded_text = preprocess_file(file)
        # Perform plagiarism check with the database
        plagiarism_results = check_plagiarism_with_database(uploaded_text)
        return render(request, 'plagiarism_result.html', {'plagiarism_results': plagiarism_results})
    return render(request, 'home.html')

def plagiarism_result(request):
    # Access plagiarism results from the view function (if needed)
    plagiarism_results = request.context.get('plagiarism_results')
    return render(request, 'plagiarism_result.html', {'plagiarism_results': plagiarism_results})
def check_plagiarism_with_database(uploaded_text):
    plagiarism_results = []
    # Iterate through all files in the database
    for database_file in FileRepository.objects.all():
        reference_text = preprocess_file(database_file.file)
        # Calculate cosine similarity for plagiarism detection
        similarity_score = calculate_similarity(uploaded_text, reference_text)
        # Set a threshold for plagiarism (adjust as needed)
        plagiarism_threshold = 0.75  # Example threshold, adjust based on your needs
        if similarity_score >= plagiarism_threshold:
        # Convert cosine similarity to percentage (multiply by 100)
          similarity_percent = similarity_score * 100
      # Format similarity score with four decimal places
          formatted_score = "{:.4f}".format(similarity_score)
          plagiarism_results.append({
            'catagory': database_file.title,
            'filename': database_file.file,
            'similarity_score': formatted_score,  # Use formatted score
            'similarity_percent': f"{similarity_percent:.2f}%"  # Format as percentage with 2 decimals
          })
    print(plagiarism_results)
    return plagiarism_results
def preprocess_file(file):
    """
    Extracts text from various file types (doc, docx, pdf, txt) for plagiarism checking.
    Args:
        file: The uploaded file object.


    Returns:
        str: The preprocessed text content of the file.
    """
    text = ''
    filename, file_extension = os.path.splitext(file.name)
    if file_extension in ['.doc', '.docx']:
        # Read doc/docx files
        document = Document(file)
        for paragraph in document.paragraphs:
            text += paragraph.text.strip()  # Remove leading/trailing whitespace
    elif file_extension == '.pdf':
        # Read PDF files
        pdf_reader = PyPDF2.PdfReader(file)
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text().strip()  # Remove leading/trailing whitespace
    elif file_extension == '.txt':
        # Read text files
        text = file.read().decode('utf-8').strip()  # Read as UTF-8 and remove whitespace
    else:
        # Handle unsupported file types (raise an exception or log a warning)
        raise ValueError(f"Unsupported file type: {file_extension}")
    # Additional preprocessing steps can be added here (e.g., lowercase conversion, stop word removal)
    return text
def calculate_similarity(text1, text2):
    """
    Calculates cosine similarity between two text strings using CountVectorizer and cosine_similarity.
    Args:
        text1: The first text string.
        text2: The second text string.
    Returns:
        float: The cosine similarity score between the two texts.
    """
    vectorizer = CountVectorizer()
    vectorized_text = vectorizer.fit_transform([text1, text2])
    similarity = cosine_similarity(vectorized_text)[0][1]
    return similarity
```

2. **Detector app (models.py):**

```
from django.db import models
# Create your models here.
class FileRepository(models.Model):
    catagory = models.CharField(max_length=255, blank=True)
    author = models.CharField(max_length=255, blank=True)
    file = models.FileField(upload_to='uploads/')
    departement = models.CharField(max_length=255, blank=True)
    college = models.CharField(max_length=255, blank=True)
    degree = models.CharField(max_length=255, blank=True)
    status = models.CharField(max_length=255, blank=True)
```

3. **Authenticate app (views.py):**

```
from audioop import reverse
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import login,logout
from django import forms
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render
from .forms import FeedbackForm
from detector.views import coordinator,user
class RegistrationForm(UserCreationForm):
    email = forms.EmailField(required=True)
    class Meta:
        model = User
        fields = ["username", "email", "password1", "password2"]
# def registration_view(request):
#     if request.method == "POST":
#         form = RegistrationForm(request.POST)
#         if form.is_valid():
#             user = form.save()
#             login(request, user)
#             return redirect("index")
#     else:
#         form = RegistrationForm()

#     return render(request, "registration/register.html", {"form": form})
def index(request):
    return render(request, "registration/login.html")
    # if request.user.is_authenticated:
    #     return redirect(profile_view)
    # else:
    #     return redirect(logout_view)
    # return render(request, "index.html")
def profile_view(request):
    # Fetch the user's profile data
    if request.user.is_staff:
        if request.user.is_superuser:
          return redirect('admin:index')
        # User is a staff member
        # return render(request, 'registration/second.html')
        return redirect(coordinator)
    else:
        # User is not a staff member
        return redirect(user)
        # return render(request, 'registration/profile.html')
def logout_view(request):
    logout(request)
    return render(request, "registration/login.html")
    # return redirect('login')  # Replace 'login' with the name of your login page URL
def confirm_registration(request, user_id):
    user = get_object_or_404(User, id=user_id)
    # Perform any necessary logic to confirm the registration
    # For example, you can update a field on the user object to mark it as confirmed
    user.is_active = True
    user.save()
    # Render a template to display a confirmation message to the user
    return render(request, 'registration/confirmation.html')
# views.py
def feedback_view(request):
    if request.method == 'POST':
        form = FeedbackForm(request.POST)
        if form.is_valid():
            form.save()
return render(request, 'feedback.html', {'form': form, 'success_message': 'Thank you for your feedback!'})
    else:
        form = FeedbackForm()
    return render(request, 'feedback.html', {'form': form})
```

4. **Template or UI (base.html):**

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Research Document Repository</title>
  <!-- Include Bootstrap CSS -->
  <link                                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <style>
    /* Custom styles for the header and footer */
    .navbar {
      background-color: #007bff;
    }
    .navbar a {
      color: #fff;
    }
    .custom-box {
      background-color: #f8f9fa;
      padding: 40px;
      border-radius: 10px;
      text-align: center;
    }
    #main {
      margin-top: 60px;
      margin-bottom: 100px;
      padding-bottom: 70px;
    }
    .footer {
      background-color: #f8f9fa;
      padding: 10px 0;
      text-align: center;
      position: fixed; /* Change position to relative */
      bottom: 0;
      width: 100%;
      z-index: 999;
    }
    .center-div {
      margin-left: auto;
      margin-right: auto;
    }
    .custom-list {
      max-height: 300px; /* Set maximum height for scrollability */
      overflow-y: auto; /* Enable vertical scrolling */
      margin: 0 auto; /* Center the list horizontally */
      text-align: center; /* Center the list content */
      padding-left: 80px; /* Add left padding */
      padding-right: 80px; /* Add right padding */
    }
    h1 {
      text-align: center;
      font-size: 24px;
      color: #337ab7;
      margin-bottom: 20px;
    }
    p {
      /* padding: 20px 0; */
      text-align: center;
      font-size: 14px;
```

```
            color: #777;
        }
        /* Additional styles for the footer */
        .footer p {
            font-size: 14px;
            color: #777;
        }
        .footer a {
            margin: 0 10px;
        }
    </style>
</head>
<body>
    <!-- Header with navigation links -->
    <nav class="navbar navbar-expand-lg navbar-dark">
        <div class="container">
            <a class="navbar-brand" href="#">Document Repository</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/about/">About</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/contact/">Contact</a>
                </li>
            </ul>
        </div>
    </div>
  </nav>
  <!-- Main content -->
  {% block content %}
  {% endblock %}
</body>
  <!-- Footer with social media links and copyright -->
  <footer class="footer">
      <div class="container">
          <p>This system allows researchers and students at Haramaya University to upload, store, and share their research and project documents. It also integrates a plagiarism detection tool to ensure the originality of uploaded content.</p>
          <p>© 2024 Research Repository. All rights reserved.</p>
          <!-- Social media icons (replace with actual links) -->
          <a href="#" class="primary">
              <i class="bi bi-facebook"></i> Facebook
          </a>
          <a href="#" class="info">
              <i class="fab fa-twitter"></i> Twitter
          </a>
      </div>
  </footer>
  <!-- Include Bootstrap JS (optional) -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <script>
    // Update the label when a file is selected
    $('#fileInput').on('change', function() {
      var fileName = $(this).val().split('\\').pop(); // Get the file name
      $(this).next('.custom-file-label').html(fileName); // Update the label
    });
/////////////
    document.getElementById("myForm").addEventListener("submit", function(event) {
    var fileInput = document.getElementById("fileInput");
    if (fileInput.files.length === 0) {
      event.preventDefault(); // Prevent form submission
      alert("Please select a file");
    }
  });
  </script>
</html>
```

4.4. Testing

4.4.1. Unit Testing

detector.views

- ✓ Test upload functionality to ensure files are saved correctly.
- ✓ Verify the preprocess_file function handles different file types (DOC,DOCX, PDF, TXT) and extracts text accurately.
- ✓ Isolate and test the calculate_similarity function to confirm it calculates cosine similarity between text strings effectively.

- ✓ Write unit tests for the check_plagiarism_with_database function. Simulate different scenarios with varying similarity scores and ensure plagiarism results are generated correctly.

authenticate.views

- ✓ Test user registration flow using the RegistrationForm. Validate form fields (username, email, password) are handled as expected.
- ✓ Write unit tests for the profile_view function to ensure it redirects users based on their staff and superuser status.
- ✓ Test login and logout functionality using the login and logout functions from django.contrib.auth.

4.4.2. System Testing

detector.views

- ✓ Manually upload different file types (DOC, DOCX, PDF, and TXT) and verify they are processed correctly.
- ✓ Test plagiarism detection by uploading a file and comparing results with expected outcomes.
- ✓ Simulate multiple users accessing the system concurrently to evaluate performance under load.

authenticate.views

- ✓ Perform user registration using the registration form, ensuring successful account creation.
- ✓ Test user login and logout functionality through the user interface. Validate error handling for invalid credentials.
- ✓ Simulate different user roles (student, researcher, and coordinator) and verify they are directed to appropriate views.

4.4.3. Acceptance Testing

- ✓ Involve potential users (students, researchers, coordinators) to test the system and gather feedback.

- ✓ Present users with mockups or prototypes to validate if the system meets their needs and expectations.
- ✓ Conduct user acceptance testing (UAT) after development to ensure the system functions as intended from a user perspective.

4.4.4. Performance Testing

- ✓ Measure response times, system load, and resource utilization under various user loads and file upload scenarios.
- ✓ Identify potential bottlenecks and optimize the system for efficient performance.

Test Cases for Web-Based Research and Project Documents Repository and Plagiarism Detector System

Test ID	Description	Expected Result
Unit Testing		
UT-1	detector.views - Upload functionality	Uploading a valid file (DOC, DOCX, PDF, and TXT) saves the file correctly in the database.
UT-2	detector.views - preprocess_file function	The function handles different file types (DOC, DOCX, PDF, and TXT) and extracts the text content accurately.
UT-3	detector.views - calculate_similarity function	The function calculates the cosine similarity between two text strings correctly.
UT-4	detector.views - check_plagiarism_with_database function	The function analyzes uploaded file text against database and generates plagiarism results based on predefined similarity thresholds. Test various scenarios with different similarity scores and ensure accurate results (e.g., high similarity indicates plagiarism).

UT-5	authenticate.views - User registration	Registration form validates user input (username, email, password) and creates a new user account upon successful registration.
UT-6	authenticate.views - profile_view function	The function redirects users based on their staff and superuser status (e.g., staff to staff view, superuser to admin view).
UT-7	authenticate.views - Login and Logout functionality	Users can successfully log in with valid credentials and log out using the provided functions. Login fails with appropriate error messages for invalid credentials.
System Testing		
ST-1	detector.views - Manual file upload	Uploading different file types (DOC,DOCX, PDF, TXT) manually processes the files correctly without errors.
ST-2	detector.views - Plagiarism detection	Uploading a file with known plagiarism content (e.g., containing text from existing database entries) generates plagiarism detection results that match expected outcomes (e.g., high similarity score indicates plagiarism).
ST-3	detector.views - System performance under load	Simulating multiple users accessing the system concurrently doesn't cause significant performance degradation or system crashes.
ST-4	authenticate.views - User registration through form	Users can successfully register using the registration form, creating new accounts within the system.

ST-5	authenticate.views - User login and logout via UI	Users can log in and log out through the user interface. Login fails with informative error messages for incorrect credentials.
ST-6	authenticate.views - User role access control	Different user roles (student, researcher, coordinator) are directed to appropriate views within the system based on their assigned permissions.
Acceptance Testing		
AT-1	User involvement	Potential users (students, researchers, coordinators) provide feedback on the system's usability, functionality, and suitability for their needs based on mockups or prototypes.
AT-2	User Acceptance Testing (UAT)	During UAT, users confirm that the system meets their expectations and performs as intended for document repository, plagiarism detection, and other functionalities.
Performance Testing		
PT-1	Performance under load	Measure system response times, load capacity, and resource utilization under various user loads and file upload scenarios. Identify potential bottlenecks and optimize the system for efficient performance across different usage patterns.

Table 25 Test Cases for Web-Based Research and Project Documents Repository and Plagiarism Detector System

4.4. Clients Feedback

Sample feedback for "Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University" project

- The user interface is clean and well-organized, making it easy to navigate through different sections and features.
- The account creation process is straightforward, and users can easily update their account information. Additionally, it would be helpful to provide users with a self-service password recovery option, allowing them to reset their passwords independently without having to rely on contacting administrators. This would enhance user convenience and reduce administrative workload.
- Uploading and downloading research papers and projects is hassle-free, and the version control feature is useful.
- The search functionality is fast and allows users to find relevant documents based on different criteria, such as title, author, or keywords.
- The color scheme and font choice could be improved for better readability.
- The system should provide more advanced document organization options, such as the ability to create folders or tags for better categorization
- It would be beneficial to have more advanced search filters, such as filtering by document type or specific date ranges.

4.5. User Manuals

1. Getting Started:
   - ✓ To access the system, your account needs to be created by the administrator. Once your account is created, you will receive login credentials.
   - ✓ Open the system login page and enter your username and password to log in.

2. Account Management:
   - ✓ After logging in, you can view and update your account information.
   - ✓ To change your password, go to the "Account Settings" section, select the "Change Password" option, and follow the prompts.
   - ✓ If you encounter any issues with your account, contact the administrator for assistance.

3. Document Management:
   - ✓ To upload research papers and projects, click on the "Upload" button or similar option.
   - ✓ Follow the instructions to select the files from your computer and provide relevant details, such as title, author, and keywords.
   - ✓ Once uploaded, the documents will be available in the system for further processing.
4. Downloading Documents:
   - ✓ Researchers and students can download documents from the system for reference or collaboration purposes.
   - ✓ However, before downloading, the Coordinator needs to approve the document.
   - ✓ Once the Coordinator approves a document, it becomes accessible for download.
5. Plagiarism Detection:
   - ✓ The system provides a plagiarism detection feature to analyze the submitted documents for potential plagiarism.
   - ✓ To submit a document for plagiarism detection, follow the instructions provided in the system.
   - ✓ Once the analysis is complete, you will receive a plagiarism detection report.
6. Security and Privacy:
   - ✓ The system ensures the security and privacy of your academic materials.
   - ✓ Avoid sharing your login credentials with others and keep your password confidential.
   - ✓ If you suspect any security issues, report them to the administrator immediately.
7. Troubleshooting and Support:
   - ✓ For any technical issues or questions, If you still need assistance, contact the system's technical support or help desk.

4.6. Conclusion

The development of the Web-Based Research and Project Documents Repository and Plagiarism Detector System for Haramaya University has been successfully completed. The system provides researchers and students with an efficient platform to upload, download, and manage research papers and projects. Additionally, the plagiarism detection feature enhances academic integrity by identifying potential instances of plagiarism. The system's user interface is clean and well-organized, making it easy to navigate and utilize its various features.

Based on client feedback, the system has received positive reviews regarding its usability, account management, document management, and search functionality. However, there are areas that can be improved to further enhance the user experience.

4.7. Recommendation

- Password Recovery Option: Implement a self-service password recovery option to allow users to reset their passwords independently. This will improve convenience for users and reduce the workload on administrators.
- Readability and Design: Address the feedback regarding the color scheme and font choice to improve readability. Consider using user-friendly colors and fonts to enhance the overall aesthetics of the system.
- Document Organization: Enhance the system's document organization capabilities by introducing features such as folders or tags. This will allow users to categorize and manage their documents more effectively.
- Advanced Search Filters: Expand the search functionality by incorporating more advanced search filters. Users should be able to filter documents by type (e.g., research paper, project), specific date ranges, or other relevant criteria. This will help users find the desired documents more efficiently.
- Ongoing Support: Provide ongoing technical support to users to address any issues or questions they may have. Establish a dedicated support channel or help desk to ensure prompt assistance.
- User Training and Education: Develop comprehensive user manuals and training materials to assist users in utilizing the system effectively. This will enhance user adoption and satisfaction.
- Continuous Improvement: Seek regular feedback from users and stakeholders to gather insights for further system enhancements and updates. Consider conducting periodic user acceptance testing (UAT) to ensure the system meets evolving user needs.

By implementing these recommendations and continuously improving the system based on user feedback, the Web-Based Research and Project Documents Repository and Plagiarism Detector

System can become an indispensable tool for researchers, students, and coordinators at Haramaya University, fostering a culture of academic excellence and integrity.

Reference

1. Mohabey, N., et al., *PLAGIARISM DETECTION FOR PROJECT REPORT USING MACHINE LEARNING.*
2. Ali, A. and A.Y. Taqa, *Analytical Study of Traditional and Intelligent Textual Plagiarism Detection Approaches.* Journal of Education and Science, 2022. **31**(1): p. 8-25.
3. Wikipedia. *"Class-responsibility-collaboration card"*. 2023 [cited Decmber 13, 2023; Available from: https://en.wikipedia.org/wiki/Class-responsibility-collaboration_card.
4. Altexsoft. *"Functional and Nonfonctional Requirements*. 2023 [cited 13/12/2023; Available from: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types.
5. SYSTEMS, S. *Use Case Modeling*. 2023 [cited December 13, 2023; Available from: https://sparxsystems.com/resources/tutorials/uml/use-case-model.html.
6. Modeling, A. *Essential Use Case Modeling*. 2023 [cited December 13, 2023; Available from: https://agilemodeling.com/artifacts/essentialusecase.htm.
7. LucidChart. *System Use Case Modeling*. 2023 [cited December 13, 2023; Available from: https://www.lucidchart.com/pages/uml-use-case-diagram.
8. hotjar. *User Interface Prototyping*. 2023; Available from: https://www.hotjar.com/ui-design/glossary/prototype/.
9. Ambysoft. *User Interface Layer*. 2023; Available from: https://ambysoft.com/essays/classTypeArchitecture.UserInterfaceLayer.
10. Ambysoft. *Controller/Process Layer*. 2023 [cited 2023 December 25]; Available from: https://ambysoft.com/essays/classTypeArchitecture.ProcessLayer.
11. Ambysoft. *Bussiness/Domain Layer*. 2023; Available from: https://ambysoft.com/essays/classTypeArchitecture.DomainLayer.
12. Ambysoft. *Persistence Layer*. 2023; Available from: https://ambysoft.com/essays/classTypeArchitecture.PersistenceLayer.
13. Wikipedia. *Class Diagram*. 2023; Available from: https://en.wikipedia.org/wiki/Class_diagram.
14. Paradigm, V. *Sequence Diagram*. 2023; Available from: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram.

15. VisualParadigm. *Activity Diagram*. 2023; Available from: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram.

16. GeeksforGeeks. *State Chart Diagram*. 2023; Available from: https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/.

17. GeeksforGeeks. *Entity Relation Model*. 2023; Available from: https://www.geeksforgeeks.org/introduction-of-er-model/.

18. freeCodeCamp. *Normalizatio of Tables*. 2023; Available from: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples.

19. Torres, A., R. Galante, and M.S. Pimenta. *Towards a uml profile for model-driven object-relational mapping*. in *2009 XXIII Brazilian Symposium on Software Engineering*. 2009. IEEE.

20. Emadi, S. and F. Shams. *From UML component diagram to an executable model based on Petri Nets*. in *2008 International Symposium on Information Technology*. 2008. IEEE.

21. Lincke, S.J., T.H. Knautz, and M.D. Lowery. *Designing system security with UML misuse deployment diagrams*. in *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*. 2012. IEEE.