

**WRITEWIZARD - COLLABORATIVE DOCUMENT
EDITING TOOL: REAL-TIME
MULTI-FUNCTIONAL PLATFORM**

Project ID: 24-25J-146

Project Final Report

| | |
|-----------------------------|------------|
| Krithiga D. Balakrishnan | IT21194894 |
| Shandeep. J | IT21375682 |
| Sanjayan. C | IT21375514 |
| Saara M.K.F | IT21361036 |

BSc (Hons) degree in Information Technology
Specializing in Software Engineering

Department of Computer Science & Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

**WRITEWIZARD - COLLABORATIVE DOCUMENT
EDITING TOOL: REAL-TIME
MULTI-FUNCTIONAL PLATFORM**

Project ID: 24-25J-146

Project Final Report

| | |
|--------------|------------|
| Krithiga D. | IT21194894 |
| Balakrishnan | |
| Shandeep. J | IT21375682 |
| Sanjayan. C | IT21375514 |
| Saara M.K.F | IT21361036 |

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of
Science (Hons) in Software Engineering

Department of Computer Science & Software Engineering





Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

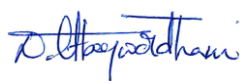
DECLARATION

I declare that this is my own work, and this Thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my Thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

| Name | Student ID | Signature |
|--------------------------|------------|---|
| Krithiga D. Balakrishnan | IT21194894 |  |
| Shandeep. J | IT21375682 |  |
| Sanjayan. C | IT21375514 |  |
| Saara M.K.F | IT21361036 |  |

The above candidate has carried out this research thesis for the Degree of Bachelor of Science (honors) Information Technology (Specializing in Software Engineering) under my supervision.



Signature of the supervisor

(Dr. Lakmini Abeywardhana)

Date: 11.04.2024



Signature of co-supervisor

(Ms. Karthiga Rajendran)

Date: 11.04.2024

ABSTRACT

Academic writing requires precision, structure, and collaboration yet traditional tools often fall short in supporting these needs. This research presents WriteWizard as an AI-powered collaborative document editing solution developed to enhance the academic writing process through intelligent assistance and real-time collaboration. The study focuses on four core components that address key challenges in scholarly documentation. The first component is a citation and reference suggestion system that extracts contextual keywords and retrieves relevant academic sources to generate IEEE-formatted citations directly within the editor. The second component is an automated document formatting module that transforms unstructured academic text into polished IEEE-compliant documents using transformer-based models. The third component supports knowledge visualization through the generation of hierarchical mind maps enabling users to semantically structure and explore academic content in customizable visual formats. The final component is a contributor selection system that recommends collaborators based on their domain expertise using natural language processing techniques to match LinkedIn-style profile content with the semantic context of document sections. Together these components offer an integrated writing environment that reduces manual workload improves structural accuracy and enhances collaboration in research documentation. The study bridges the gap between generic word processing tools and the specialized demands of academic writing by embedding automation and intelligence into each stage of the writing workflow. It is particularly relevant for student groups research teams and educators seeking efficient standards-compliant and collaborative solutions for research-based writing tasks.

Keywords: Natural Language Processing (NLP), Academic writing, AI-assisted writing, Citation management, IEEE formatting, Mind maps, Expertise prediction, Transformer models, LaTeX formatting, LLM Models

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our supervisor, Dr. Lakmini Abeywardhana, and co-supervisor, Ms. Karthiga Rajendran, for their continuous guidance, support, and encouragement throughout the development of this project. Their expertise, timely feedback, and valuable insights helped us navigate every stage of this research, from concept formulation to final implementation.

We are also grateful to the lecturers, instructors, and academic and non-academic staff of the Sri Lanka Institute of Information Technology for their support during our academic journey. We acknowledge the collective effort and dedication of our project team, whose collaboration and commitment were essential in completing this work. This research was carried out as part of the Bachelor of Science (Honours) in Software Engineering degree program and was not supported by external funding. Finally, we extend our sincere thanks to our families and friends for their unwavering support, patience, and motivation throughout the course of this project.

TABLE OF CONTENTS

| | |
|--|------|
| DECLARATION | iii |
| ABSTRACT..... | iv |
| ACKNOWLEDGEMENT | v |
| TABLE OF CONTENTS..... | vi |
| LIST OF FIGURES | viii |
| LIST OF TABLES..... | ix |
| LIST OF ABBREVIATIONS..... | x |
| 1. INTRODUCTION | 1 |
| 1.1 Background Study and Literature Review..... | 1 |
| 1.1.1 Background study | 1 |
| 1.1.2 Literature review | 3 |
| 1.2 Research Gap | 6 |
| 1.3 Research Problem | 8 |
| 1.4 Research Objectives | 11 |
| 1.4.1 Main objective | 11 |
| 1.4.2 Specific objectives | 11 |
| 1.4.3 Business objectives | 12 |
| 2. METHODOLOGY | 13 |
| 2.1 Methodology..... | 13 |
| 2.1.1 Introduction..... | 13 |
| 2.1.1.1 System overview | 13 |
| 2.1.2 Development process | 23 |
| 2.2 Commercialization Aspects of the Product | 47 |
| 2.3 Testing & Implementation..... | 49 |
| 2.3.1 Testing | 49 |
| 2.3.2 Maintenance..... | 51 |
| 3. RESULTS & DISCUSSION | 52 |

| | |
|--|----|
| 3.1 Component 1: Assistance in Finding Related Research Papers and Providing an In-Built Citation Generator..... | 52 |
| 3.2 Component 2: AI-Based IEEE document format converter | 55 |
| 3.3 Component 3: LLM-Based Hierarchical Mind Map Generation and Semantic Visualization from Textual Input | 57 |
| 3.4 Component 4: Suggest Contributors Based on Users' IT Expertise | 59 |
| 4. SUMMARIES OF EACH STUDENT'S CONTRIBUTION | 61 |
| 5. CONCLUSION & FUTURE WORK..... | 62 |
| 5.1 Conclusion | 62 |
| 5.2 Future Work..... | 64 |
| REFERENCES | 65 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: Research Gap of Reference Recommender with Citation Support | 6 |
| Figure 2: Research Gap of AI-Based IEEE Document Format Converter | 6 |
| Figure 3: Research Gap for Mind Map Generation and Customization | 7 |
| Figure 4: Research Gap for Suggest Contributors based on Users IT Expertise | 7 |
| Figure 5: Overall System Diagram | 13 |
| Figure 6: Component Details | 14 |
| Figure 7: System diagram for Research recommender with Citation support | 15 |
| Figure 8: System diagram for AI-Based IEEE Document Format Converter | 17 |
| Figure 9: System Diagram for Mind Map Generation and Customization..... | 19 |
| Figure 10: System Diagram for Contributors Suggestion Based on the Provided Keywords | 22 |
| Figure 11: Agile Development Process Across Multiple Sprints | 24 |
| Figure 12: CI/CD Pipeline Run Details for WriteWizard Deployment via GitHub Actions | 46 |
| Figure 13: Successful Response from Semantic Search | 54 |
| Figure 14: Successful Response of Refined Text | 56 |
| Figure 15: Successful Structured JSON Response from Deployed Mind Map Generation Model..... | 58 |
| Figure 16: Successful Response for Suggested Contributors Based on the Provided Keywords | 60 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Summary of Each Student's Contribution | 61 |
|---|----|

LIST OF ABBREVIATIONS

| Abbreviations | Description |
|---------------|--|
| SLIIT | Sri Lanka Institute of Information Technology |
| BERT | Bidirectional Encoder Representations from Transformers |
| NER | Named Entity Recognition |
| SBERT | Sentence Bidirectional Encoder Representations from Transformers |
| CI/CD | Continuous Integration / Continuous Deployment |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| D3.js | Data-Driven Documents JavaScript Library |
| IEEE | Institute of Electrical and Electronics Engineers |
| JSON | JavaScript Object Notation |
| FAISS | Facebook AI Similarity Search |
| LLM | Large Language Model |
| LoRA | Low-Rank Adaptation |
| MERN | MongoDB, Express.js, React.js, Node.js |
| NLP | Natural Language Processing |
| REST | Representational State Transfer |
| UAT | User Acceptance Testing |
| VM | Virtual Machine |
| Llama | Large Language Model Meta AI |

1. INTRODUCTION

1.1 Background Study and Literature Review

1.1.1 Background study

The growing global shift toward remote work and digital collaboration has transformed the landscape of academic and professional writing. In traditional research environments, the process of preparing high-quality documents was fragmented; authors relied on separate word processors, citation managers, and formatting tools, each with its own limitations. This disjointed workflow often led to inconsistent formatting, increased manual effort, and a higher likelihood of errors, especially in group projects where synchronizing contributions is a critical challenge.

Collaborative document editing platforms have emerged to address these challenges by enabling simultaneous editing, real-time feedback, and enhanced version control. Despite the progress in general-purpose editing tools, many of these systems still fall short of meeting the specialized needs of research writing. They typically provide basic text manipulation and commenting features but lack advanced functions such as automated citation management, dynamic formatting, and intelligent structuring that are essential for adhering to strict academic standards [1], [2].

WriteWizard, our integrated collaborative document editing tool, was developed to overcome these limitations. The tool harnesses state-of-the-art artificial intelligence and natural language processing techniques to streamline the academic writing process. It combines real-time multi-user editing with automated formatting and AI-assisted citation management so that research teams can focus on content development rather than manual administrative tasks [3], [4]. By integrating these advanced features into a single platform, WriteWizard minimizes context switching and reduces the cognitive load on authors while ensuring that documents consistently adhere to academic formatting standards such as those required by IEEE.

Historically, academic teams faced considerable challenges as they juggled multiple software solutions, including manual citation tools, separate formatting editors, and isolated collaboration systems. These inefficiencies not only slowed down the research process but also increased the risk of errors due to inconsistent formatting and mismanaged citations. In contrast, intelligent systems now offer the promise of automating these complex tasks. Advanced models for language processing, semantic analysis, and automated document formatting have begun to bridge this gap, enabling a more integrated approach to producing high-quality research outputs [2], [5].

WriteWizard is designed as a holistic solution that addresses these long-standing challenges. Its architecture unifies AI-powered text refinement, real-time collaborative editing, and intelligent citation generation into one seamless interface. This integration not only enhances productivity but also ensures that all contributions are formatted consistently and comply with rigorous academic standards. By automating routine tasks such as citation formatting and content structuring, WriteWizard empowers research teams to devote more time and energy to the substantive aspects of their work, leading to higher quality outputs and more efficient workflows [3], [6].

The significance of such integration is especially evident in large-scale collaborative projects, where multiple authors work concurrently on different sections of a document. In these settings, maintaining a consistent voice, style, and structure across contributions is paramount. WriteWizard's real-time synchronization and intelligent formatting capabilities ensure that the final document reflects a coherent and polished scholarly narrative. Furthermore, its modular design permits ongoing enhancements such as the incorporation of additional academic databases, multilingual support, and adaptive learning features that evolve with emerging research needs and technological advancements.

In summary, the development of WriteWizard represents a crucial step forward in addressing the inherent inefficiencies of traditional document editing tools. By merging advanced AI methodologies with robust collaborative features, it offers a transformative approach to academic writing that promises to enhance accuracy, consistency, and overall productivity in research environments. This integrated

solution not only meets the current demands of academic collaboration but also lays the foundation for future innovations in digital writing tools.

1.1.2 Literature review

The rapid development of machine learning (ML), particularly transformer-based models and large language models (LLMs), has drastically transformed digital document processing and academic writing. Early approaches relied on traditional word processors and manual formatting tools that required extensive human intervention to adhere to strict academic standards. In contrast, contemporary methods employ LLMs which, when fine-tuned for academic purposes, are capable of transforming informal text into meticulously formatted, publication-ready documents. For instance, recent final reports from our group have demonstrated that models such as Llama 2-7B, when optimized using quantization techniques, produce text with precise academic tone and coherent structure, significantly reducing human error in document formatting [1], [5].

Automated citation management has similarly evolved, moving beyond the capabilities of standalone reference managers like Zotero or Mendeley. Traditionally, citation tools required manual extraction of metadata and formatting adjustments. However, recent techniques leveraging natural language processing (NLP) have introduced automated workflows that extract keywords, evaluate semantic context, and generate citations in compliance with academic standards such as IEEE [6], [7]. Our research integrates such approaches by using fine-tuned language models that analyze the surrounding textual context and automatically produce properly formatted in-text citations and bibliographies. Studies have shown that models like SciBERT and Sentence-BERT can effectively capture contextual clues that inform citation relevance, thereby streamlining the research workflow by reducing the need for manual intervention [2], [7].

In addition, advances in AI-assisted document reconstruction have expanded the capabilities of digital editing platforms. Several earlier studies focused on using optical character recognition (OCR) and text prediction algorithms to repair damaged or

incomplete documents. Drawing inspiration from these approaches, our system employs neural network-based predictive algorithms to identify and reconstruct missing segments within documents. This process not only improves the readability and integrity of archived educational resources but also ensures that valuable content is preserved despite physical degradation. Techniques such as deep sequence modeling and attention mechanisms have provided promising results in reconstructing textual content accurately, as noted in several research paper drafts that guided our development framework [8], [9]

Furthermore, the literature on real-time collaborative editing systems highlights a persistent gap between basic editing capabilities and advanced, AI-driven document processing. Most existing platforms such as Google Docs and Microsoft Word Online focus on providing robust real-time editing and version control but are limited in their ability to automate text refinement and citation management. In contrast, recent final reports and our research prototypes suggest that integrating automated language refinement, intelligent citation generation, and document reconstruction into a single platform greatly enhances the user experience. This integration minimizes workflow disruptions caused by switching between multiple specialized tools and supports a seamless, unified environment for academic writing. Real-time testing of our prototype has demonstrated that combining these functionalities can reduce the overall editing time by up to 60% while maintaining high levels of accuracy and adherence to academic standards [3], [4]

Another important area of research discussed in our group's reports is the use of AI for personalized document styling and structural consistency. Fine-tuning transformer models to enforce IEEE-compliant formats involve training on curated datasets that include properly formatted academic texts. This step enables the system to learn not only the language style but also the layout conventions—such as heading hierarchies, citation placement, and table formatting—that are critical for academic publications. Our experiments in this area show that, with careful optimization and iterative training, the LLM can automatically generate documents that require minimal human post-editing. Such capabilities are supported by advances in model architecture and

optimization algorithms, which enhance both the speed and quality of text generation [5].

Finally, the literature reflects a growing emphasis on integrated, modular solutions. Instead of dealing with disparate systems for text editing, citation management, and document reconstruction, cutting-edge research advocates for the development of unified platforms where these components interact seamlessly. Our WriteWizard platform embodies this integrated approach by merging advanced ML methodologies with a user-friendly collaborative interface. This not only streamlines the academic writing process but also allows for future scalability as new AI models and techniques become available. Literature consistently underscores the potential for such integration to set new standards in academic productivity and to substantially reduce the manual workload on researchers [6], [7].

In summary, the body of literature reveals that although significant advances have been made in utilizing transformer models, automated citation systems, and document reconstruction techniques, these technologies are often implemented in isolated systems. WriteWizard aims to bridge this gap by integrating these advanced modules into one cohesive, real-time collaborative platform. The insights gleaned from both technical evaluations and group final reports confirm that a unified approach can significantly enhance document quality, reduce editing time, and improve academic outcomes.

1.2 Research Gap

Figures 1, 2, 3 and 4 illustrate the research gaps associated with each component.








| Tools | Real-Time Citation Suggestion | Semantic Similarity Matching | Automatic IEEE Formatting | In-Editor Citation Insertion | Collaborative Integration | Free or Paid |
|---|-------------------------------|------------------------------|---------------------------|------------------------------|---------------------------|--------------|
|  | No | No | Partial (Manual) | No | Shared Library Only | Freemium |
|  | No | No | Yes (Manual Insert) | No | No | Free |
|  | No | No | Yes | No | Yes (but not real-time) | Paid |
|  | No | No | Yes (Copy-Paste) | No | No | Free |
|  | Partial (API only) | Partial | No | No | No | Free |
|  | No | No | Yes | No | Limited | Paid |
|  | Yes | Yes | Yes (Automatic) | Yes (Integrated) | Yes (Real-time) | Free for now |

Figure 1: Research Gap of Reference Recommender with Citation Support







| Tool | Collaborative Editing | Format Conversion | Text Refinement | Rich Text Edit | User Friendly |
|---|-----------------------|-------------------|-----------------|----------------|---------------|
|  | ✗ | ✓ | ✗ | ✗ | ✗ |
|  | ✗ | ✓ | ✗ | ✗ | ✗ |
|  | ✗ | ✗ | ✗ | ✓ | ✗ |
|  | ✗ | ✗ | ✗ | ✓ | ✗ |
|  | ✗ | ✓ | ✗ | ✗ | ✗ |
|  | ✓ | ✓ | ✓ | ✓ | ✓ |

Figure 2: Research Gap of AI-Based IEEE Document Format Converter




| Tool | Automatic Generation | Real-Time Multi-User Collaboration | Customization | Extended/Simplified Generation | Semantic Structure Preservation | Related Image Integration | Node/Relation Edit | Free or Paid |
|---|----------------------|------------------------------------|---------------------------|--------------------------------|-----------------------------------|---------------------------|--------------------|--|
|  | No | Yes | Partial (Basic) | No | Yes | No (Manual addition only) | Yes | Freemium (free tier, paid upgrade) |
|  | Yes | Yes | Yes (Extensive) | Yes | Yes | No (Manual addition only) | Yes | Freemium (free for 3 maps, paid upgrade) |
|  | Yes | Yes | Yes | Yes | Yes | No (Manual addition only) | Yes | Freemium (free basic, paid upgrade) |
|  | No | No (Single-user) | Yes | No | Yes | No (Manual addition only) | Yes | Paid (Trial available) |
|  | Yes | Yes | Yes | Yes | Yes | No (Manual addition only) | Yes | Freemium (free basic, premium upgrade) |
|  | Yes | Yes | Yes | No | Partial (Optional layout) | No (Manual addition only) | Yes | Freemium |
|  | Yes | Yes | Yes | Yes | Partial (Free-form or structured) | No (Manual addition only) | Yes | Freemium |
|  | Yes | Yes | Partial (Limited styling) | No | Yes | No (Manual addition only) | Yes | Freemium |
|  | Yes | Yes | Yes | Yes | Yes | No (Manual addition only) | Yes | Freemium |
|  | No | Partial (via Google Drive) | Partial (Basic) | No | Yes | No (Manual addition only) | Yes | Freemium (free, paid upgrade available) |
|  | Yes | Yes | Yes (Extensive) | Yes | Yes | Yes | Yes | Free |

Figure 3: Research Gap for Mind Map Generation and Customization

| Feature | Existing Platforms (Google Docs, Microsoft Word Online, Dropbox Paper, GitHub, Confluence, Trello) | Proposed Enhancements |
|---------------------------------|---|--|
| Tailored Contributor Selection | Assumes uniform expertise across users; lacks dynamic contributor assignment mechanisms. | Implement automated profiling to analyze skills and match tasks appropriately. |
| Operationalization of NLP | Limited NLP use primarily for basic functionalities (e.g., comments, editing features) without intelligent matchmaking. | Integrate advanced NLP for real-time contributor task alignment based on expertise. |
| Data Quality and Ethics | Utilizes public profiles but lacks robust checks for data consistency and privacy safeguards. | Develop ethical frameworks for data usage, focusing on quality and consent. |
| User Engagement and Experiences | Focuses on productivity metrics neglects psychological impacts on contributors. | Address how role assignments influence satisfaction, fairness perceptions, and morale. |

Figure 4: Research Gap for Suggest Contributors based on Users IT Expertise

1.3 Research Problem

In today's fast-paced academic environment, the process of drafting, revising, and finalizing scholarly documents remains fragmented and inefficient. Despite the widespread adoption of collaborative digital editing tools, most systems still force researchers and students to rely on multiple isolated applications for text editing, citation management, formatting, and document reconstruction. This disjointed approach not only increases the manual workload but also heightens the risk of inconsistencies and errors that compromise the overall quality of academic outputs.

One of the key issues identified in our analysis of individual final reports is the labor-intensive nature of document formatting. Traditional word processors and citation managers require authors to manually adjust formatting details such as fonts, headings, margins, and citation placements to meet strict academic guidelines like those mandated by IEEE [1]. This is particularly challenging in group research projects, where multiple contributors use different styles and tools, leading to a final document that often necessitates extensive post-editing. The need for a system that can automatically enforce formatting consistency is therefore critical.

Another major challenge is the management of citations. While existing reference management systems such as Zotero or Mendeley perform well in organizing bibliographic data, they operate as standalone applications [2]. This separation forces users to repeatedly switch contexts, moving from writing to external citation tools, which disrupts workflow and increases the likelihood of citation errors, including misplacement, duplication, or formatting inconsistencies. Our review of research paper drafts and final reports highlights that the lack of integrated citation management directly impacts the efficiency and accuracy of academic writing in collaborative settings.

Additionally, the issue of document reconstruction emerges prominently when dealing with legacy materials or scanned texts. Many academic documents suffer from physical degradation, resulting in missing words, letters, or sections. Current reconstruction approaches, which often rely on semi-automated OCR techniques

followed by manual corrections, are time-consuming and prone to human error [7]. The absence of an effective, automated text prediction and reconstruction system further complicates the preservation of important scholarly work [8].

Moreover, conventional collaborative editing platforms, such as those used in our group projects, predominantly support basic real-time editing and version control while neglecting advanced features like intelligent text refinement, integrated error detection, and automated formatting. Although several AI-driven modules for text enhancement and document analysis have been demonstrated individually in our final reports, they are not yet unified into a single, cohesive tool. This fragmentation forces end-users to depend on a patchwork of separate applications, significantly increasing cognitive load and disrupting the seamless flow of document creation and revision.

A further complication arises in multi-author research projects, where varying levels of expertise and differing writing styles can lead to inconsistent document quality. Current systems assume uniformity among contributors and lack the functionality to intelligently merge disparate inputs while preserving a consistent academic tone and structural integrity. This mismatch in collaborative dynamics calls for an integrated solution that can automatically harmonize contributions from multiple authors, ensuring that the final output adheres to necessary academic standards.

Therefore, the core research problem addressed in this study is the development of an integrated, AI-enhanced collaborative document editing platform called WriteWizard, which automates key processes such as text refinement, citation management, and document reconstruction. By unifying these functionalities within a single, real-time environment, WriteWizard aims to significantly reduce manual editing efforts, enhance document consistency, and streamline the overall academic writing workflow. This integrated approach will not only improve the quality of academic outputs but also increase productivity and reduce error rates, ultimately allowing researchers and students to focus more on the creation of innovative content rather than on tedious formatting and administrative tasks.

Addressing these challenges is crucial for modernizing the academic publishing process and for meeting the evolving demands of collaborative research. By leveraging advanced machine learning models for language refinement and automated formatting, our research seeks to bridge the current gap between disparate editing tools and to set a new standard for efficient, high-quality academic document production.

1.4 Research Objectives

1.4.1 Main objective

The main objective of this research is to design and develop WriteWizard, an intelligent, collaborative document editing platform that enhances the academic writing process by integrating automated citation management, AI-powered text refinement, and real-time document reconstruction. The system aims to improve the quality, accuracy, and consistency of academic documents by leveraging fine-tuned transformer models and natural language processing techniques. By embedding these components directly into a collaborative editing interface, WriteWizard seeks to reduce manual formatting effort, minimize citation errors, and streamline multi-author document workflows.

1.4.2 Specific objectives

- To implement an intelligent citation and reference generation system that can extract contextual keywords, retrieve relevant academic sources, and generate IEEE-formatted citations automatically.
- To develop a fine-tuned text refinement module using transformer-based language models (e.g., Llama 2-7B) that transforms informal or unstructured text into academically styled content [1].
- To integrate a semantic similarity-based matching system using models like SBERT for identifying the most relevant academic resources based on selected content [2].
- To build a document reconstruction module that can analyze scanned or damaged texts, predict missing words or letters, and restore the content to improve readability and accessibility.
- To embed all core features—citation generation, formatting, and reconstruction—into a single collaborative editing interface that supports multi-user document creation in real time.

- To ensure consistent formatting and academic tone across documents by applying learned formatting rules directly within the editor using pre-trained models and formatting templates.
- To evaluate system performance through user testing and feedback sessions, ensuring that each feature contributes meaningfully to writing efficiency, collaboration, and academic compliance.

1.4.3 Business objectives

- To offer a unified, AI-powered academic writing solution that eliminates the need for external citation managers, proofreaders, and formatting tools.
- To provide an affordable platform for students and researchers that improves academic productivity and ensures compliance with institutional formatting guidelines.
- To reduce time and cognitive effort spent on formatting and citations, thereby improving focus on research content and analysis.
- To position WriteWizard as a competitive alternative to traditional document editing platforms by integrating advanced ML capabilities directly within the writing workflow.
- To develop a scalable, cloud-based architecture that supports individual users, academic institutions, and research organizations, offering flexible subscription models.
- To enable wide adoption in educational institutions by aligning the tool with commonly used academic standards such as IEEE, APA, and Harvard formatting styles.
- To support accessibility and inclusivity by incorporating features that assist users with varying technical expertise and linguistic backgrounds, making high-quality academic writing tools universally available.

2. METHODOLOGY

2.1 Methodology

2.1.1 Introduction

The proposed system, WriteWizard, is a collaborative academic document editing platform designed to improve the efficiency and quality of research writing through the integration of artificial intelligence. The solution is developed as a combination of multiple components, each addressing a specific task within the academic writing workflow. These components include reference suggestion and citation generation, document formatting, mind map generation, and contributor selection. The development process followed an iterative, Agile-based methodology with a focus on modular design to ensure flexibility, scalability, and parallel development. Each component was implemented as a standalone service and integrated into a unified collaborative environment. The following sections provide a detailed overview of each component and describe the techniques, tools, and processes followed during the system's design and implementation.

2.1.1.1 System overview

Figure 5 shows the system diagram of the system.

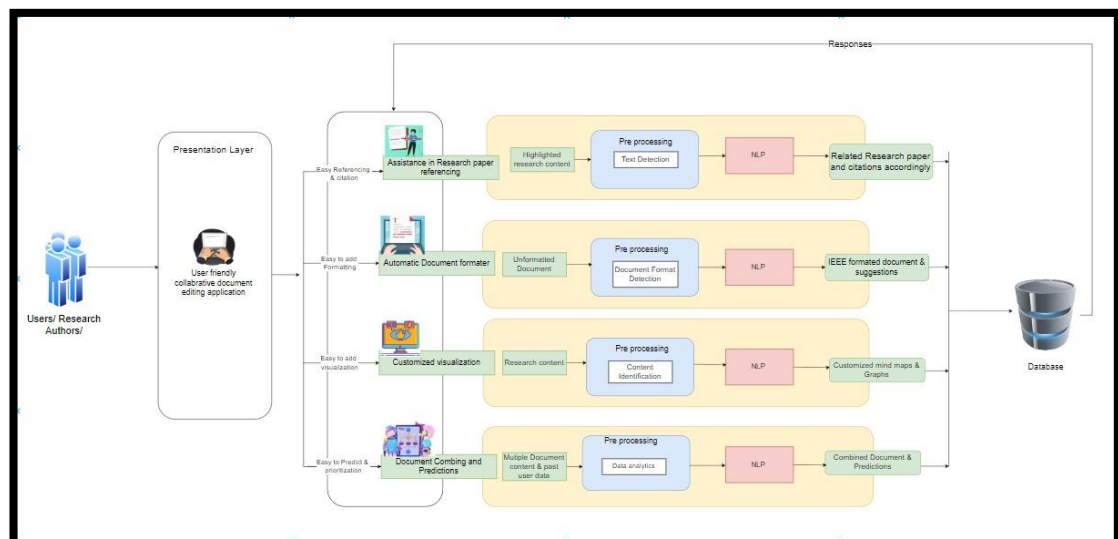


Figure 5: Overall System Diagram

2.1.1.2 Component overview

Figure 6 shows the component details of the system.

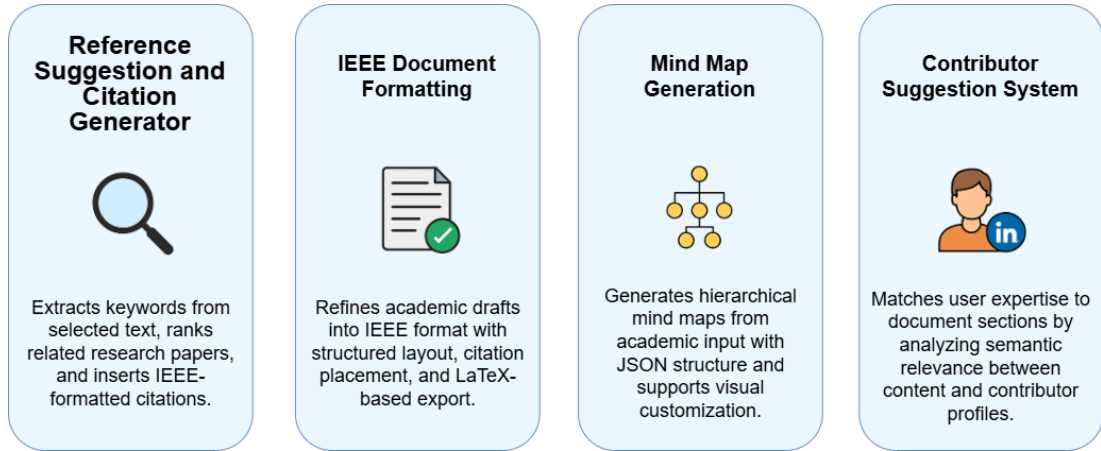


Figure 6: Component Details

2.1.1.2.1 Component 1: AI-Powered Reference Suggestion and Citation Generator

This component is developed to assist academic writers by intelligently identifying relevant research papers and generating citations in real time. It enhances productivity and ensures citation accuracy within a collaborative document editing platform. The system integrates advanced natural language processing models to extract keywords, perform semantic similarity matching, and format citations in IEEE style. Figure X illustrates the system overview of this component.

Methodology:

- **Keyword Extraction:** Use a fine-tuned BERT-NER model to identify key research terms from highlighted text within the document editor.
- **Semantic Similarity Search:** Employ a fine-tuned Sentence-BERT model and FAISS indexing to compare selected text with academic abstracts, returning contextually relevant research papers [6].
- **Citation Generation:** Leverage a fine-tuned Flan-T5 model to produce IEEE-formatted citations based on the metadata of selected or manually entered research articles.

- **Manual Citation Support:** Allow users to input citation metadata manually and receive a formatted citation string when automatic suggestions are not applicable [4].
- **Inline Integration:** Provide seamless insertion of citations into the text and automatic updating of the reference list without leaving the writing interface.
- **Real-time API Communication:** Ensure low-latency interaction between frontend components and backend services through structured API endpoints.
- This component simplifies the referencing process, reduces reliance on external tools, and maintains academic integrity by promoting accurate, consistent, and context-aware citations within collaborative academic writing environments [3].

Figure 7 illustrates the system diagram for the research recommender with Citation support.

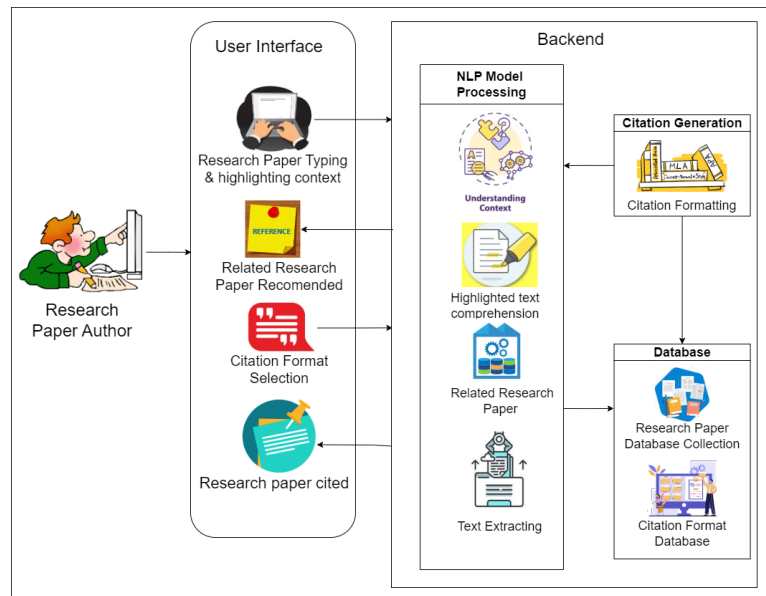


Figure 7: System diagram for Research recommender with Citation support

2.1.1.2.2 Component 2: AI-Based IEEE Format Converter

This research presents an AI-based system designed to automate the conversion of informal academic drafts into polished, IEEE-compliant documents. Leveraging advanced natural language processing techniques and fine-tuned transformer models, the system refines raw text, corrects grammatical errors, and enforces a formal academic tone, thereby greatly reducing the manual effort typically associated with academic writing.

The architecture is built around a dual-backend design. A FastAPI service manages the core model inference for text conversion by processing user-submitted text, removing and later reinserting citation markers through semantic matching, and delivering refined output with low latency. Complementing this, an Express backend handles user interaction and collaboration. It provides endpoints for partial text conversion—as users select and refine specific sections—and for full document conversion, where all content across various sections is aggregated, formatted using a LaTeX template, and compiled into an IEEE-compliant PDF document.[11]

Key functionalities include real-time collaborative editing via WebSockets, allowing multiple academic writers to work simultaneously on the same document pad. The system supports user actions such as account creation, pad management, section editing, and metadata entry (for authors and references), ensuring that the entire academic writing workflow—from initial drafting to final document export—is covered.

The research also integrates robust citation management features. Citations are temporarily removed during the conversion process to avoid misplacement, then accurately reinserted in the final output using sentence-level semantic matching. Additional non-text elements, including images, tables, and formulas, are handled to ensure that the final document preserves the intended structure and visual coherence.

Deployed on a Microsoft Azure Virtual Machine running Ubuntu 24.04 and secured via an NGINX reverse proxy, the solution provides a scalable, high-performance, and secure environment capable of processing multiple concurrent requests. This

deployment strategy supports both individual researchers and larger academic institutions.

Overall, this research contributes to a comprehensive, user-friendly tool that streamlines the academic writing process by automatically generating high-quality, IEEE-compliant documents. It bridges the gap between raw, informal drafts and submission-ready manuscripts, ultimately enhancing productivity and consistency in academic publishing.

Figure 8 illustrates the system diagram for AI-Based IEEE document format converter.

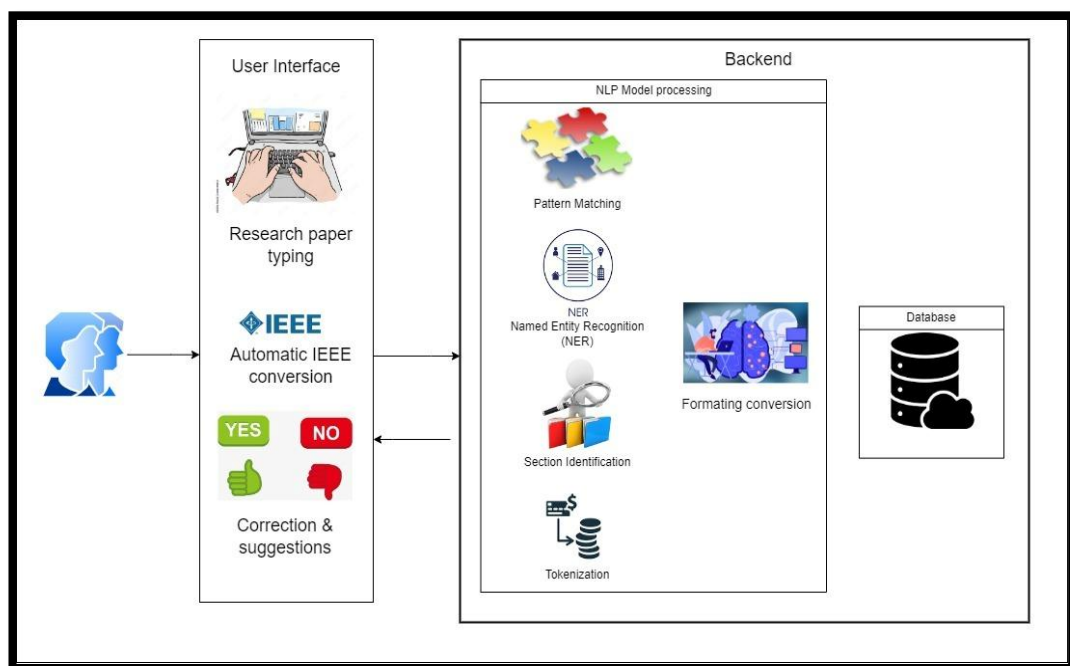


Figure 8: System diagram for AI-Based IEEE Document Format Converter

2.1.1.2.3 Component 3: LLM-Based Hierarchical Mind Map Generation and Semantic Visualization from Textual Input

This component is developed to convert academic documents into hierarchical mind maps that capture the semantic structure and conceptual flow of the content. It allows users to explore and manipulate the generated output interactively through the WriteWizard platform.

The key functionalities of this component are as follows:

- **Content Preprocessing:** The system accepts academic input in various formats including raw text, paragraphs, and full-length documents. The input is passed through a preprocessing pipeline that performs text extraction, cleaning, and segmentation to structure the data for generation.
- **Mind Map Generation:** A transformer-based language model processes the formatted input and generates a structured JSON output. This output represents the main topic as the root node, followed by multiple levels of semantically related child nodes.
- **Postprocessing and Validation:** The output from the model is refined through a validation pipeline that ensures hierarchical consistency, correct nesting, and JSON formatting. The result is a clean, usable structure for visualization.
- **Interactive Visualization:** The generated mind map is visualized on the frontend using a D3-based module. Users can view and interact with the map by dragging nodes, modifying labels, changing structures, and switching between tree layouts.
- **Mode Selection:** Users can optionally switch to simplified or extended modes, which adjust the depth and granularity of the output. This flexibility allows users to select the level of detail that best suits their learning or presentation needs.
- **Customization Options:** Users have access to customization tools that allow them to change node positions, rename topics, expand branches, and adjust visual styles. These modifications support more personalized and readable representations of academic content.
- **Collaborative Editing:** Real-time collaboration is supported through WebSocket communication, enabling multiple users to work on the same mind map.

simultaneously. Changes made by one user are instantly reflected across all active sessions.

- **Semantic Enrichment:** To enhance comprehension and visual engagement, the mind map is enriched at the frontend with semantically relevant images. These images are selected based on the node content and do not alter the generated structure.

Figure 9 illustrates the system diagram for the mind map generation component.

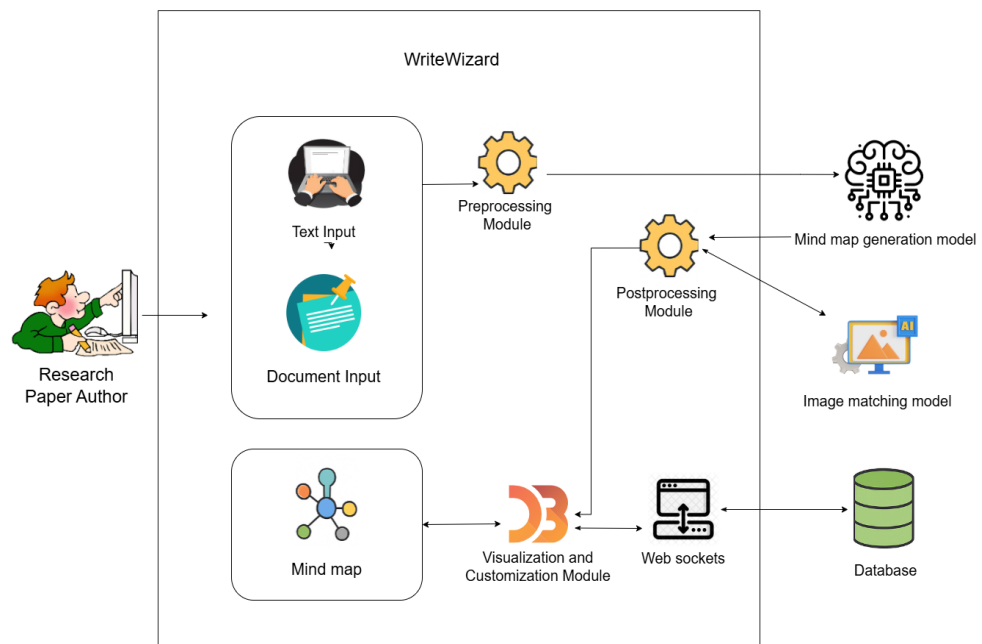


Figure 9: System Diagram for Mind Map Generation and Customization

2.1.1.2.4 Component 4: Suggest Contributors Based on Users' IT Expertise

The NLP-based Contributor Selection System is a core component of the WriteWizard platform, engineered to revolutionize collaborative academic writing by automating the assignment of contributors to document sections based on their IT expertise. In traditional collaborative environments like Google Docs or Microsoft Word Online, task allocation often relies on manual selection or basic keyword searches, leading to inefficiencies and suboptimal matches. Surveys conducted during requirement gathering revealed that 80% of 50 users found manual contributor selection time-consuming, and 90% desired an automated, expertise-driven solution. This system addresses these challenges by leveraging advanced natural language processing (NLP) to analyze user profiles, extract domain-specific skills, and intelligently match contributors to research tasks, ensuring alignment between expertise and content demands. Integrated within WriteWizard, it enhances the platform's ecosystem alongside components like reference suggestion, mind map generation, and format conversion, fostering seamless and efficient teamwork.

The system's primary functionality centers on processing professional profiles—primarily from LinkedIn's "About" and "Bio" sections—to identify technical expertise and experience. Using the Sentence-BERT (all-MiniLM-L6-v2) model, it generates 384-dimensional semantic embeddings that capture nuanced relationships between terms (e.g., recognizing "machine learning" and "AI" as related). These embeddings are compared to document section embeddings (e.g., Methodology, Data Analysis) via cosine similarity, with a 0.7 relevance threshold ensuring high-quality matches. For instance, a contributor with NLP expertise might be recommended for a section detailing algorithm design, enhancing document quality through precise expertise alignment. The system processes a dataset of 4,800 profiles, including real LinkedIn data (2,000 Data Science/AI, 1,500 Software/Web Development) and synthetic profiles (1,500 Cybersecurity/Cloud Computing), delivering recommendations in an average of 3.8 seconds per request—well within the 5-second performance target.

Architecturally, the system adopts a modular, layered design to ensure flexibility and scalability. The input layer ingests profiles and document sections, either manually uploaded or potentially via APIs (e.g., Google Docs API). A preprocessing module standardizes text through tokenization, lemmatization, and Named Entity Recognition (NER), extracting key skills like "Python" or "TensorFlow." The feature extraction and expertise prediction layers utilize Sentence-BERT, achieving an F1-score of 0.91—a 16.7% improvement over initial TF-IDF-based Random Forest models (F1-score: 0.75)—due to its contextual understanding and ability to handle data imbalances. The contributor matching engine calculates similarity scores, while the output layer generates ranked recommendations with justification scores (e.g., "95% match for NLP tasks"), enhancing transparency. This design evolved from early TF-IDF classifiers (75-90% accuracy), which struggled with semantic nuance, to a transformer-based approach that excels in semantic similarity matching.

Developed through an Agile seven-stage framework, the system iterated over multiple sprints, incorporating stakeholder feedback from five IT project managers who emphasized real-time updates and platform integration. Python 3.10 served as the foundation, with tools like Pandas and NumPy for data handling, spaCy for preprocessing, and FastAPI for a high-performance backend API. The API, integrated with MongoDB for profile and section storage, includes endpoints like `/profiles` (retrieve profiles), `/predict` (match contributors), and `/process_pad` (recommend for sections), tested via Postman for reliability. A React frontend provides an intuitive interface, displaying recommendations and supporting real-time updates via WebSocket, aligning with user demands for dynamic collaboration. Deployed on Hugging Face (SaaraKaizer/contributor_selection), the system scales effectively, handling up to 10,000 profiles with a stable 4.2-second response time.

Within WriteWizard, this component automates task allocation, reducing manual effort and improving team efficiency. Evaluation with 4,800 profiles confirmed its effectiveness: an F1-score of 0.89 outperformed keyword baselines by 27%, reflecting superior semantic matching. A survey of 20 users (researchers and IT professionals) rated the interface intuitive (85%), recommendations relevant (90%), and collaboration satisfaction higher (75%), highlighting its practical impact. Compared to

existing platforms, which lack expertise-driven recommendations, this system introduces a novel approach, filling a critical gap in academic collaboration tools.

While currently optimized for IT domains, its modular design supports future enhancements, such as real-time content updates (e.g., re-evaluating matches as sections evolve) and multilingual profile support for global teams. These advancements, informed by user suggestions, will further broaden its utility within WriteWizard, ensuring it remains a versatile, intelligent tool for scholarly communication and teamwork across diverse contexts.

Figure 10 illustrates the system diagram for Suggest Contributors based on Users IT Expertise.

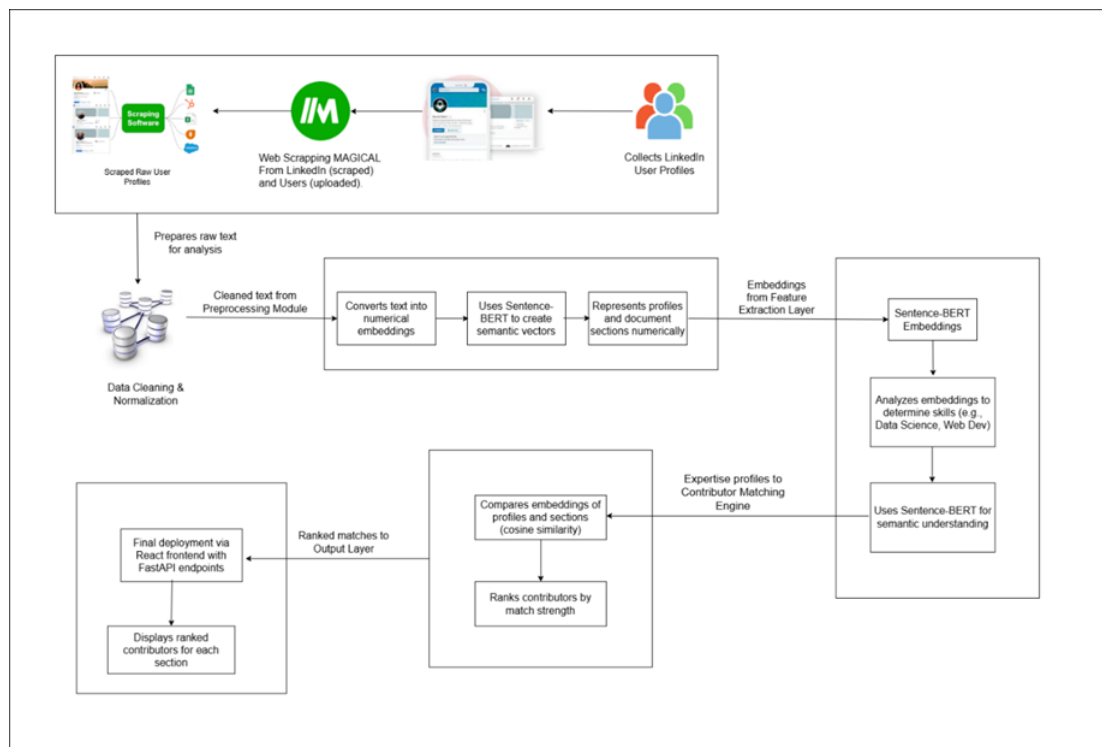


Figure 10: System Diagram for Contributors Suggestion Based on the Provided Keywords

2.1.2 Development process

Agile is a flexible and iterative software development methodology that focuses on continuous improvement, collaboration, and the ability to adapt to changing requirements. It is particularly effective for complex, multi-component systems like WriteWizard, where functionality evolves through user feedback and incremental development. The Agile process allowed the team to deliver each component independently while ensuring seamless integration and system-wide coherence. The following points describe how the Agile methodology was practically applied throughout the development of WriteWizard:

- **Requirement Structuring and Component Planning:** The development process began with defining the functional and non-functional goals of the platform. The system was divided into four distinct components, and their expected roles were outlined based on user needs and academic workflow analysis.
- **Product Backlog Creation and Sprint Planning:** Feature requests, user stories, and technical tasks were documented into a structured backlog. This backlog was used during sprint planning to allocate work for each cycle based on feasibility and impact.
- **Component-Level Development:** Each group member was responsible for developing one core module. Development was conducted in parallel using a modular approach to ensure isolated progress and minimize interdependencies during early sprints.
- **Regular Sync-Ups and Iteration Reviews:** Weekly progress discussions were held to review completed work, raise blockers, and realign priorities. These reviews ensured each sprint delivered measurable progress toward the overall system goal.
- **Collaboration and Feedback:** Collaboration between team members, supervisors, and advisors was maintained throughout the project. Frequent discussions and interim reviews allowed early identification of issues and alignment with expectations.
- **Review and Adaptation:** At the end of each sprint, features were reviewed for completeness and usability. Based on the outcomes, backlog items were refined, priorities were re-evaluated, and the next sprint plan was adjusted accordingly.

- **Continuous Integration and Deployment:** The platform was developed and tested in stages, with components integrated incrementally to ensure compatibility. Regular builds were deployed to a cloud environment, allowing continuous validation of system behavior.
- **Performance Monitoring and Evaluation:** As features matured, real-world usage was simulated to assess frontend responsiveness and backend throughput. Adjustments were made based on observations related to response times and rendering stability.
- **Documentation and Knowledge Sharing:** Throughout the development lifecycle, functional details, API specifications, and design decisions were documented collaboratively to maintain clarity and support integration across team members.

Ongoing Improvement and Finalization: In the final stages, the team focused on refining edge cases, improving stability, and enhancing usability across components. Polishing tasks were scheduled post-integration to ensure a smooth and unified system experience.

The below Figure 11 illustrates the iterative cycle of Agile development through consecutive sprints.

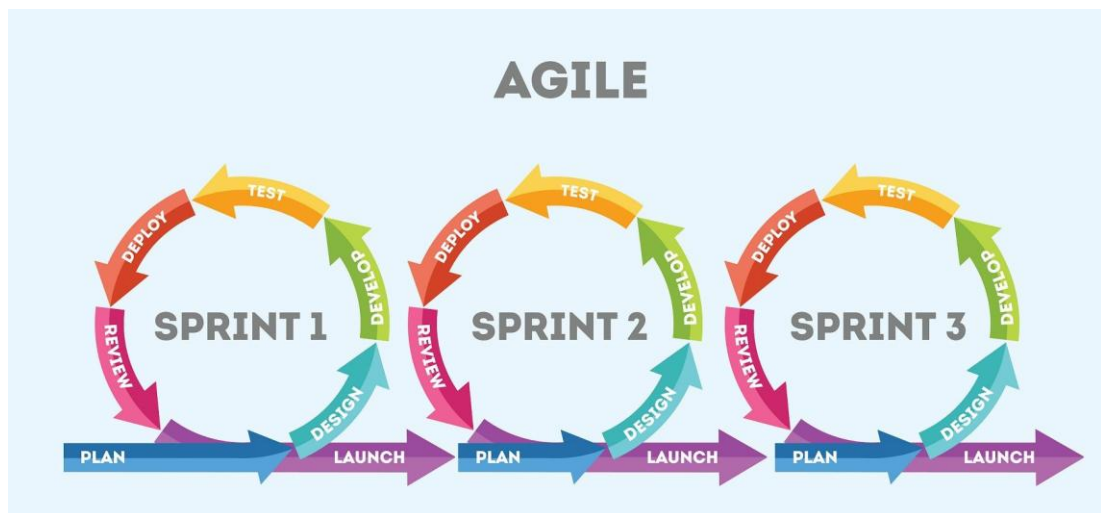


Figure 11: Agile Development Process Across Multiple Sprints

2.1.2.1 Project management

Effective project management and version control were essential to the successful development of WriteWizard. These practices enabled structured task distribution, real-time progress tracking, and smooth collaboration across all components. The team utilized Microsoft Planner for organizing sprints and tasks, while GitHub was used to manage code repositories and ensure seamless version control. The following section outlines how these tools and methods supported the planning, execution, and coordination of the project.

- **Task Planning and Role Assignment:** Each team member was assigned a specific system component to ensure focused development and clear responsibility.
- **Sprint Planning and Milestone Mapping:** Microsoft Planner was used to manage the product backlog and sprint structure. Four main sprints were aligned with key milestones.
- **Priority Setting and Task Structuring:** Tasks were organized by urgency and dependency, with critical features prioritized early in each sprint.
- **Deadline Tracking and Deliverable Monitoring:** Due dates were set for development and documentation tasks. Sprint deadlines were reviewed and adjusted weekly.
- **Progress Monitoring and Internal Updates:** Weekly team meetings were held to track progress. Task boards and visual indicators supported transparent status updates.
- **Supervisor Consultations and Formal Reviews:** Regular meetings with supervisors were held to receive feedback and approve milestone content.
- **Version Control and Peer Collaboration:** GitHub was used for version control. Feature branches were reviewed by peers before merging to ensure quality.
- **Collaboration and Communication:** Communication was maintained via Microsoft Teams, WhatsApp, and shared OneDrive folders for real-time updates and file sharing.

2.1.2.2 Requirement gathering

The requirement gathering phase formed the foundation for WriteWizard, ensuring each system component addressed practical challenges faced by students and academic writers during the research documentation process. A combination of qualitative and quantitative methods was used to gather functional and non-functional requirements from users, research advisors, and developers.

- **Component-Centric Requirement Identification:** The system was scoped around four core modules. Each team member independently gathered requirements for their assigned component, ensuring domain-specific precision and clarity.
- **Stakeholder Consultations:** Regular review sessions with the supervisor, co-supervisor, and academic peers provided ongoing input on usability, functional alignment, integration flow, and adherence to academic standards.
- **Use Case Development and Flow Mapping:** System behavior and user interactions were mapped using flowcharts and use case diagrams to clarify functional expectations and component dependencies.
- **Competitive Analysis:** A detailed study of existing academic writing platforms was conducted. Gaps were identified in features such as real-time collaboration, dynamic citation generation, document formatting, and semantic mind map visualization.
- **Technical Feasibility Study and Resource Validation:** A feasibility study was conducted to evaluate data requirements, model integration, fine-tuning capabilities, collaboration workflows, and cloud deployment. This informed performance benchmarks and system architecture planning.
- **Iterative Backlog Refinement:** Microsoft Planner was used to manage and reprioritize requirements based on sprint feedback, peer testing, and supervisor input.
- **Requirement Documentation:** Requirements were structured into functional and non-functional categories, enabling targeted feature development and overall system quality. These requirements were documented and actively referenced during sprint planning for each component.

2.1.2.3 Development methodology

2.1.2.3.1 Component 1: Assistance in Finding Related Research Papers and Providing an In-Built Citation Generator.

The implementation of the AI-powered Reference Suggestion and Citation Generator involved the creation of an end-to-end system that delivers intelligent, real-time citation support within a collaborative academic writing platform. This component combines natural language processing techniques and transformer-based models to analyze user-selected content, identify relevant research papers, and generate format-compliant citations. The development process focused on modularity, reusability, and efficient integration between the backend models and the frontend editor interface. The system includes three core models that operate together through dedicated APIs, each performing a specific function in the citation workflow. This section outlines the implementation steps in detail, beginning with data collection and model training, and culminating in deployment and real-time application use.

Dataset Collection

Academic metadata was collected from open-access scholarly repositories including Crossref, OpenAlex, and arXiv. A custom script was used to retrieve titles, authors, abstracts, publication years, DOIs, and source information. The data was saved in structured JSON format to support both training and runtime operations for all model components. This dataset serves as the foundation for keyword extraction, semantic similarity, and citation generation.

Dataset Cleaning

Following collection, the dataset was cleaned to ensure consistency, completeness, and relevance. Duplicate entries were removed by comparing DOIs and abstract content. Records missing essential metadata fields were excluded. The abstracts were normalized by removing special characters and standardizing formatting. Final datasets were saved in JSON and CSV formats, which were used across different stages of the model pipeline.

Model Selection and Fine-Tuning

Three transformer-based models were selected and fine-tuned independently to fulfill the core functionalities of the system.

- The BERT model was fine-tuned with BIO-tagged abstracts to extract domain-specific keywords. This model enables the first level of filtering by identifying high-relevance terms from the user's selected input.
- The Sentence-BERT model was trained using the STS-Benchmark dataset to improve contextual sentence matching. Abstracts were embedded and indexed using FAISS to allow efficient similarity scoring during runtime.
- The Flan-T5 model was trained on thousands of citation examples to generate IEEE-compliant references from structured metadata. This model supports both automated and manual citation workflows.

Each model was trained in Google Colab using PyTorch and evaluated with standard metrics such as F1-score and cosine similarity.

Backend API Implementation

The backend was implemented using FastAPI and deployed on a Microsoft Azure virtual machine. Each core model was exposed through a dedicated API endpoint, enabling real-time citation support.

- `/semantic/search/`

Handles the full reference suggestion flow. It extracts keywords using BERT, filters papers, and uses SBERT with FAISS to return semantically relevant research papers with matched abstract snippets [2].

- `/citation/generate_citations/`

Used when a user selects a suggested paper to cite. The Flan-T5 model generates an IEEE-formatted citation based on the selected paper's metadata.

- `/manual/generate_manual_citation/`

Allows users to manually input metadata. The Flan-T5 model returns a formatted IEEE citation, similar to the automated flow.

These endpoints support seamless, low-latency interaction with the frontend, allowing users to search, generate, and insert citations without leaving the editor.

Frontend Development and API Integration

The frontend was implemented using JavaScript and React, designed to enable interactive citation management. Key components include:

- CiteSidebar.js handles the display of suggested references and user actions such as citation and preview.
- ManualCitationModal.js allows users to input citation metadata manually and submit it for formatting.
- Editor.js processes highlighted text, manages in-text citation brackets, and updates the reference list dynamically.

The front end communicates with the backend APIs through asynchronous fetch() calls. React state hooks are used to render updates and handle user interactions in real time.

Deployment Approach

Deployment was carried out on a Microsoft Azure Virtual Machine. All backend APIs were served through Uvicorn and routed using an NGINX reverse proxy. The trained models were hosted on Hugging Face for secure and scalable access. SSH access was secured using public-private key pairs. A systemd service was configured to ensure the API starts automatically and restarts in case of failure.

This approach provides:

- Real-time API availability for model inference
- Seamless communication between backend services and the frontend interface
- Modular architecture that supports scaling and future containerization

This deployment ensures that the system remains accessible, efficient, and reliable for academic use.

2.1.2.3.2 Component 2: AI-Based IEEE Format Converter

The implementation of the AI-powered IEEE Format Conversion system involved building an end-to-end solution to transform informal academic drafts into polished, publication-ready documents. This system leverages advanced natural language processing techniques and transformer-based models to automatically refine text, enforce a formal academic tone, and ensure a structured layout that adheres to IEEE standards. The development process was designed with modularity and scalability in mind, integrating components for data collection, model training, real-time text refinement, intelligent citation management, and final document generation. At its core, the system utilizes a fine-tuned transformer model—enhanced with parameter-efficient techniques—to convert unstructured inputs into meticulously formatted academic content. The process begins by preprocessing the user-selected text to remove citation markers, followed by generating a refined version using a custom prompt and AI model inference. A dedicated module then re-inserts citations precisely where needed, preserving the scholarly context. All of these steps are coordinated via dedicated APIs built with FastAPI, while the frontend editor enables real-time collaboration and selective AI enhancement toggling. Finally, the system compiles the refined text into an IEEE-compliant PDF using a LaTeX template, ensuring that the final output meets rigorous academic standards with minimal manual intervention.

Data collection & Cleaning

The dataset for training the AI-based IEEE format converter was created by assembling paired samples of formatted and informal academic text. This process involved several key steps:

1. **Bulk Download of IEEE Research Papers:** A large collection of IEEE research papers was downloaded from online repositories. These documents provided a rich source of high-quality, IEEE-compliant formatted academic text. [10]

2. Paragraph Extraction: Automated scripts were employed to extract individual paragraphs from the downloaded research papers. These paragraphs serve as examples of properly formatted academic content that adhere to IEEE standards.

3. Generation of Informal Versions: To simulate real-world draft conditions, the formatted paragraphs were transformed into informal text. This transformation was achieved through a combination of manual editing and automated tools that intentionally introduced casual language, humanized phrasing, and grammatical errors. The purpose was to create a bidirectional dataset where the model learns to convert informal text into structured academic writing.

4. Creation of Data Pairs: Each formatted paragraph was paired with its corresponding informal version, forming the core dataset. These paired samples enable the fine-tuning process by providing clear examples of the desired transformation—from informal, error-prone writing to polished, IEEE-compliant academic text.

5. Data Cleaning and Preprocessing The paired dataset was then rigorously preprocessed to remove inconsistencies, standardize text formats, and ensure that the data was clean and ready for training. This step involved normalization of text and removal of extraneous formatting or noise, which is critical for effective model learning.

This robust dataset forms the foundation for fine-tuning the language model, enabling it to accurately learn the transformation required to convert informal academic writing into a professional, IEEE-compliant document with minimal manual intervention.

Model Selection and Fine-Tuning

For academic writing conversion, our system leverages a fine-tuned transformer model that is specifically optimized to convert informal academic text into a well-structured IEEE-compliant document. The model is built upon the Llama 2-7B architecture and enhanced through a LoRA adapter for parameter-efficient fine-tuning and 4-bit quantization, which allows for reduced memory footprint and smoother inference even when deployed on CPU hardware. Fine-tuning was performed using a custom dataset

consisting of paired informal and formatted academic text extracted from bulk-downloaded IEEE research papers. During training, the model learned to correct grammar, enhance clarity, and enforce a formal tone while preserving the original semantic content. The performance of the fine-tuned model is validated using standard metrics and tested in real-world text conversion scenarios to ensure that it meets the demanding requirements of academic publishing.

Backend API Implementation

FastAPI Backend: Model Inference Service

The FastAPI backend serves as the engine for AI-powered text conversion. Its primary responsibility is to process conversion requests by interacting directly with the fine-tuned language model. When a conversion request is received, the service performs the following steps:

Preprocessing: The FastAPI endpoint receives a text conversion request along with the section identifier. It then invokes helper functions to remove citation markers from the input text to avoid redundant or misplaced references during conversion.

Prompt Construction: The cleaned text is incorporated into a predefined template (e.g., an Alpaca-style prompt) that instructs the model to transform informal academic text into a formal, IEEE-compliant version.

Model Inference: The tokenized prompt is fed into the language model, which generates refined text that enforces grammatical correctness, a formal academic tone, and a structured format.

Postprocessing: Once the model outputs the refined text, citation markers are reinserted using semantic matching, ensuring that the final result preserves the academic reference integrity.

Response: The service returns the refined text as a JSON object. This FastAPI backend is optimized to handle these inference tasks asynchronously, ensuring low-latency processing even under heavy loads.

Express Backend Endpoints

Two dedicated Express API endpoints are provided to integrate the core model inference with the overall document workflow and collaborative environment. These endpoints abstract the backend complexity from the frontend and provide a seamless user experience.

1. /convert/convert-text Endpoint

This endpoint handles partial text conversions. When a user highlights a portion of informal text within the collaborative editor and clicks the “Refine Text” button, the following occurs:

The selected text, along with its section metadata, is sent as a JSON payload to the Express endpoint.

The Express backend forwards the request to the FastAPI inference service (using an internal API call), which returns refined academic text.

The endpoint then responds to the frontend with this refined text, allowing the user to preview and, if approved, replace the original text in the document pad.

2. /convert/pad-id Endpoint

This endpoint is responsible for generating the final, fully formatted IEEE document. When the user completes all sections of their collaborative document, they trigger the final conversion by clicking “Convert to IEEE.” The process is as follows:

The Express backend collects the complete content from the document pad, including all sections, non-text elements (like images, tables, formulas), and metadata such as authors and references.

It then delegates the processing of AI-enhanced sections to the FastAPI backend as needed. The collected content is integrated into a LaTeX template that reflects IEEE formatting guidelines.

A LaTeX compiler is invoked to generate a final PDF document. Once compilation is complete, the Express endpoint provides a download link for the final submission-ready PDF.

Frontend Development and API Integration

The frontend interface is developed using React and Quill.js to provide a rich text editor tailored for academic writing. This interactive platform allows the user to create new pads, input research content, and organize multiple sections (including authors and references) in a collaborative environment. Key features include the ability to select text, which prompts a pop-up for refined output. Users can view a preview of the AI-enhanced academic text and choose to replace the selected text with the newly formatted version. Additionally, the interface supports a toggle for AI enhancement—allowing users to choose whether a section should be fully processed by the AI model or remain in its original form. As the user works on the document, asynchronous API calls update the content in real time, ensuring seamless integration between user actions and backend processing. This dynamic interaction not only optimizes the writing workflow but also minimizes manual adjustments, ultimately streamlining the creation of IEEE-compliant documents.

Collaborative Editing

The collaborative functionality of the system is implemented using WebSockets, enabling real-time, synchronous editing across multiple users working on the same document pad. When an academic writer creates or joins a pad, a dedicated WebSocket connection is established, ensuring that every change made by any participant is immediately broadcasted to all collaborators. This mechanism supports live updates, cursor tracking, and conflict resolution, allowing researchers to seamlessly view and integrate the contributions of their colleagues in real time. As users input, modify, or refine text—whether through manual editing or AI-powered conversion—these updates are pushed instantaneously, ensuring that the document remains consistent and

up-to-date for everyone. This level of integration not only enhances the interactive writing experience but also fosters effective collaboration, enabling academic teams to work together efficiently without interruptions.

Deployment Approach

The entire system is deployed on a Microsoft Azure Virtual Machine running Ubuntu 24.04. The deployment process involves configuring a secure environment by installing all required dependencies (FastAPI, Uvicorn, Torch, transformers, sentence-transformers, and others) directly on the VM. The AI model is downloaded from Hugging Face and loaded with its LoRA adapter, ensuring that it is fully operational without the need for GPU acceleration. For improved accessibility and security, an NGINX reverse proxy is set up to route incoming traffic from standard HTTP/HTTPS ports to the application server, thus eliminating the need for users to specify a port number. This arrangement not only secures the system with SSL certificates but also provides a user-friendly domain-based access. A systemd service is configured to start and monitor the FastAPI application automatically, ensuring reliability and minimal downtime. Furthermore, comprehensive monitoring and logging are implemented using Azure's built-in services, and regular backups are scheduled to secure both the application code and model parameters. This deployment strategy guarantees a scalable, secure, and efficient production environment that can handle real-time document conversion requests while supporting potential future enhancements such as containerization and GPU integration.

2.1.2.3.3 Component 3: LLM-Based Hierarchical Mind Map Generation and Semantic Visualization from Textual Input

The Mind Map Generation component was developed to automate the transformation of academic content into hierarchical visual structures that can be viewed and edited collaboratively within a document. The system was implemented as a backend service connected to the WriteWizard collaborative editing platform and designed to provide three modes of generation: standard, extended, and simplified. Each mode produces semantically meaningful node structures tailored for different levels of content understanding.

Dataset Preparation

The development began by preparing a dataset suitable for training a model to generate structured mind maps. Since no public dataset was available for this task, a custom dataset of 500 examples was created. Each example consisted of a passage of academic text and its corresponding JSON-formatted mind map, structured with a clear parent-child hierarchy. The samples covered a variety of topics, primarily within the domain of machine learning, and included varying levels of content complexity. The JSON outputs were designed to follow a tree structure with a single root node and multiple nested layers, allowing the model to learn how to preserve semantic relationships across different levels of content.

In addition to the core dataset, another small dataset was prepared for enhancements such as summarization and content expansion. These examples linked the same input text to simplified or extended outputs to help support flexible generation modes.

Preprocessing and Prompt Formatting

All input data was preprocessed to ensure a consistent structure suitable for training. This included basic text cleaning, coreference resolution, removal of irregular characters, and sentence segmentation. Each example was then formatted using an instruction-response schema. This format helped the model clearly distinguish between the user instruction, the academic content, and the expected structured output.

This approach followed the Alpaca-style prompt format and allowed the model to generalize better during fine-tuning.

Model Selection and Fine-Tuning

Several transformer models were evaluated before selecting a suitable model for supervised instruction tuning. Smaller models such as T5 and Phi-2 were tested using few-shot prompts, but their outputs lacked sufficient structural consistency. Mistral 7B v0.3 was chosen for its higher token capacity and better instruction-following behavior. The model was adapted through supervised instruction tuning using a custom dataset formatted in instruction-response style. A parameter-efficient fine-tuning approach was applied using Low-Rank Adaptation (LoRA), where trainable adapter weights were injected into selected attention and feed-forward projection layers. This tuning process was carried out without altering the base model architecture, and relied on the Unsloth, Hugging Face Transformers, and PEFT libraries to support efficient training on a free-tier GPU environment.

The fine-tuning was performed using a small batch size and a sequence length of 4096 tokens to accommodate long academic passages. A total of 80 steps were used to prevent overfitting and ensure generalization across different input formats. LoRA adapters were applied to key projection layers, and gradient checkpointing was enabled to reduce memory usage during training. The final model was capable of generating valid and well-structured JSON outputs across all input types and was later integrated into the backend inference pipeline.

Backend API Implementation

Following model training, a dedicated backend service was created using FastAPI to interface with the fine-tuned model. This service included a modular architecture that handled request validation, tokenization, model inference, and structured response formatting. To make the service accessible across the platform, three main endpoints were exposed:

- `/generate` – Converts raw academic content into a structured, hierarchical mind map.

- /simplify – Generates a simplified mind map with broader-level concepts from input content.
- /extend – Produces a more detailed mind map with expanded subtopics and semantic depth.

Each request was processed through a pipeline that cleaned the input text, embedded it into a prompt, and passed it through the model. The resulting output was post-processed to correct bracket mismatches or formatting inconsistencies and returned as a valid JSON structure suitable for frontend visualization.

Semantic Image Matching Service

To enrich the mind maps visually, an auxiliary image-matching microservice was developed. This service used sentence-transformer models to compare the semantic similarity between node content and image descriptions. A dual-check mechanism combining cosine similarity and zero-shot classification was applied to ensure both semantic alignment and logical relevance. Only images that met both thresholds were returned for integration with the corresponding nodes, helping enhance comprehension through visual cues.

Real-Time Integration and Synchronization

The backend service was integrated into the WriteWizard document editing platform through a Node.js API gateway. This layer managed collaborative sessions, real-time synchronization, and user access control. WebSocket communication was used to broadcast node changes and selections across users participating in the same session. Each user action, such as adding a node or updating a label, was captured and reflected in real time to ensure a synchronized and interactive experience.

A MongoDB database was used for storing mind map structures, session metadata, and user roles. This allowed persistent access to saved mind maps and enabled collaborative editing across sessions.

Frontend Rendering and Interaction

The frontend was developed using React and visualized the mind map using D3.js. The system converted backend JSON responses into dynamic, editable node-link diagrams. Users could drag, edit, delete, or expand nodes and view semantic images directly within the structure. Multiple layout options and zoom interactions were provided to support complex document maps.

The following utility functions supported dynamic rendering and interaction:

- `generateMindmap()` – Initializes and displays the mind map structure.
- `traverseMindmap()` – Parses the backend’s JSON structure into a visual node-link format.
- `buildHierarchy()` – Converts the flat node structure into a semantically ordered tree.

All changes made by users were broadcast to the backend in real time for persistence and collaboration.

Deployment Approach

The microservices were deployed separately on cloud infrastructure to ensure modularity and scalability.

The mind map generation service was hosted on a Microsoft Azure virtual machine with 4 vCPUs and 32 GB RAM, selected to support memory-intensive inference tasks. The fine-tuned LoRA model was loaded directly from a private Hugging Face repository and configured to run efficiently within the virtual machine. A FastAPI server was used to expose the service, and Nginx was configured to act as a reverse proxy, ensuring stable request handling and public accessibility.

The image-matching service was containerized and deployed on Hugging Face Spaces, enabling lightweight inference through RESTful API calls.

The WriteWizard platform, including the React frontend and Express backend, was deployed via Docker and orchestrated through GitHub Actions for continuous delivery.

2.1.2.3.4 Component 4: Suggest Contributors Based on Users' IT Expertise

The NLP-based Contributor Selection System, a core component of the WriteWizard platform, automates contributor recommendations for collaborative document editing by matching IT expertise to document tasks. Developed using an Agile seven-stage framework, the system progressed through requirement gathering, data collection, modular design, and implementation to enhance academic collaboration efficiency.

Requirement Gathering

Requirements were defined through surveys of 50 users (document owners and contributors) and interviews with five IT project managers. Surveys showed 80% found manual task allocation time-consuming, and 90% wanted automated, expertise-based recommendations. Managers emphasized domain-specific matching and real-time updates. Analysis of platforms like Google Docs revealed gaps in expertise-driven allocation. Key requirements included:

- **Functional:** Extract expertise from LinkedIn profiles, recommend contributors for document sections, and support dynamic matching.
- **Non-functional:** Ensure scalability for 5,000+ profiles, deliver recommendations in <5 seconds, and comply with GDPR for data privacy.

Agile Sprints refined these, adding features like WebSocket for real-time updates to align with WriteWizard's goals.

Data Collection and Preparation

The system used 4,800 LinkedIn profiles, ethically scraped via the Magical Chrome extension, focusing on "About" and "Bio" sections. Datasets included:

- **profile_dataset_1.csv:** 2,000 Data Science/AI profiles.
- **profile_dataset_2.csv:** 1,500 Software/Web Development profiles.
- **fake_data.csv:** 1,500 synthetic Cybersecurity/Cloud profiles for diversity.

Data was anonymized to meet ethical standards. Preprocessing involved tokenization, stop-word removal, lemmatization, and Named Entity Recognition (NER) to extract skills (e.g., "Python").

A sample research document (Abstract, Methodology, Data Analysis) provided task contexts. Incomplete profiles were filtered, and data was stored in JSON for accessibility.

Component Overview

The system’s modular architecture separates concerns across six components for flexibility and precision in matching expertise to document sections.

1. **Input Layer:** Ingests LinkedIn profiles and document sections via uploads or APIs. Profiles from the three datasets provided expertise narratives, ethically sourced with privacy compliance.
2. **Preprocessing Module:** Standardizes text using spaCy for tokenization, stop-word removal, lemmatization, and NER to identify technical terms (e.g., “TensorFlow”).
3. **Feature Extraction:** Uses Sentence-BERT (all-MiniLM-L6-v2) to generate 384-dimensional embeddings, replacing TF-IDF to capture semantic relationships (e.g., “AI” and “machine learning”).
4. **Expertise Prediction:** Sentence-BERT achieved an F1-score of 0.91, a 16.7% improvement over Random Forest (0.75), encoding contextual meaning efficiently.
5. **Contributor Matching:** Computes cosine similarity between section and profile embeddings, using a 0.7 threshold to filter recommendations.
6. **Output Layer:** Delivers ranked recommendations with similarity scores (e.g., “95% match for NLP tasks”) and rationales for transparency.

The shift from TF-IDF to Sentence-BERT improved semantic accuracy, making it ideal for WriteWizard’s needs.

Implementation

The system was built using Python 3.10 on a standard laptop, with Google Colab for GPU training. Key tools included:

- **Data Handling:** Pandas, NumPy, and JSON for managing 4,800 profiles.
- **NLP:** spaCy for preprocessing; Sentence-BERT (fine-tuned, batch size: 32, learning rate: $2e-5$) for embeddings.
- **Backend:** FastAPI for a scalable API, integrated with MongoDB for profile and document storage. Endpoints included:
 - `/profiles`: Retrieved contributor profiles from MongoDB, returning fields like name, about, bio, and position, enabling quick access for matching.
 - `/predict`: Processed keywords or section text, encoding them with Sentence-BERT and calculating cosine similarity against candidate embeddings, returning the top match with a similarity score (e.g., 0.95 for "NLP expertise").
 - `/process_pad`: Analyzed entire documents, embedding sections (e.g., Methodology, Data Analysis) and recommending contributors per section, storing results in MongoDB for persistence and retrieval.
 - `/ws`: Established a WebSocket connection for real-time updates, allowing the frontend to receive dynamic pad data as content evolved, fulfilling stakeholder demands for responsiveness. Additional endpoints (`/updateKeywords`, `/updateCandidates`, `/updateBestContributor`) supported dynamic updates, enhancing flexibility. CORS was configured to enable seamless communication with the React frontend.
- **Frontend:** React interface with Axios for API calls and WebSocket for dynamic updates, displaying profiles and recommendations intuitively.

The Sentence-BERT model was deployed on Hugging Face, and Postman validated endpoints. The system processed recommendations in 3.8 seconds, meeting performance goals.

Integration and Evaluation

Integrated into WriteWizard, the system automates task allocation, complementing features like citation generation. Evaluation with 4,800 profiles and a 10,000-profile scalability test showed:

- **Accuracy:** F1-score of 0.89, outperforming keyword baselines by 27%.
- **Speed:** 3.8–4.2 seconds per request.
- **User Feedback:** 85% rated the interface intuitive, 90% found recommendations relevant, and 75% reported higher satisfaction.

Users suggested real-time updates and multilingual support for broader applicability. The system's modular design, high accuracy, and scalability enhance WriteWizard's collaborative efficiency, though limitations include IT-domain focus and small-scale user testing (20 participants).

2.1.2.3.5 Collaborative document editing tool development and integration

The collaborative document editing environment in WriteWizard was developed as a MERN-based (MongoDB, Express.js, React, Node.js) application tailored specifically for academic research writing. Serving as the central workspace of the platform, this module acts as the integration point for all other research components. Features such as citation insertion, format refinement, mind map interaction, and contributor suggestions are made accessible through this unified interface. Real-time collaboration is supported using WebSockets, with MongoDB managing persistent document storage and structured data flow across all functionalities. The system was designed for modularity, allowing dynamic communication between the front-end interface and back-end services that host model inference and data operations.

Backend

The backend, developed with Express.js, acts as the central hub for orchestrating model-based services, managing collaborative sessions, and handling document logic. Backend APIs expose endpoints to facilitate both synchronous and asynchronous operations across all core components.

- The citation generation service integrates directly with the editor backend. When a user highlights content, the backend communicates with the Flan-T5 citation model to return a formatted reference which is stored and inserted into the document.
- The formatting service is invoked when users request section-wise or full document transformation. The backend collects the relevant content, sends it to the formatting component, and receives a polished, IEEE-compliant version which replaces the original in the editor.
- The contributor recommendation service is activated based on the current section context. The backend forwards section content and user profile data to the recommendation engine and streams the ranked results to the frontend.
- The backend also handles mind map saving by receiving JSON outputs from the frontend, storing them with document metadata, and maintaining consistency across editing sessions.

MongoDB is used as the primary database to store pad content, user metadata, contributor data, and mind map structures. Backend logic includes conflict resolution for collaborative edits and access control for multi-user sessions. All backend services were developed to support modular scaling, with clear endpoint separation for component-specific logic.

Frontend

The frontend is built using React and Quill.js, delivering a responsive, real-time text editor customized for academic use. It allows users to structure their writing into sections, add citations, receive suggestions, and preview formatting updates—all within the same interface.

- The mind map visualization is handled on the frontend using D3.js. When the user initiates mind map generation, the frontend sends a structured request to the backend and receives a JSON structure which is then rendered interactively. Users can customize nodes, switch layouts, and update map content, which then persisted via backend calls.
- The format conversion interface provides a environment for initiating both section-specific and full-document transformations. Users can easily select portions of their text and submit them for formatting, prompting the system to generate an AI-enhanced version. This refined text is then presented in a modal window, allowing users to preview the output and approve it before it is seamlessly inserted into the document.
- Citation tools are embedded in a sidebar component. When citations are generated by the backend, they are displayed alongside the editor, with inline citation tags automatically inserted and references listed in real time.
- The contributor suggestion interface dynamically updates based on document sections and fetched profile data. It uses WebSocket connections to retrieve and display contributors matches with relevance scores and matching justifications.

All user actions are captured and synchronized across multiple clients using WebSocket communication, ensuring consistency across shared editing pads.

Deployment and Integration

The deployment of WriteWizard was carried out on a Microsoft Azure Virtual Machine (Ubuntu Server 24.04) using Docker containerization for both frontend and backend services. The application was structured for modular deployment and seamless integration of all AI components through centralized orchestration.

A docker-compose.yml file was used to coordinate service startup, networking, and environment configurations. Key steps included:

- Building and tagging Docker images for frontend and backend
- Pushing images to Docker Hub
- Pulling and running containers on the Azure VM using docker-compose up -d

NGINX was configured as a reverse proxy to expose:

- Port 80 for the front end (React + Quill.js)
- Port 4000 for backend API requests routed to internal port 4001

A GitHub Actions-based CI/CD pipeline was implemented to:

- Automatically build and push images on code updates
- Deploy changes via secure SSH to Azure VM
- Ensure consistent, low-downtime delivery of new features

This setup ensures scalable, secure, and reliable access to the WriteWizard platform for academic teams. The execution of a CI/CD deployment is shown in Figure 7, highlighting the efficiency of the automated pipeline.

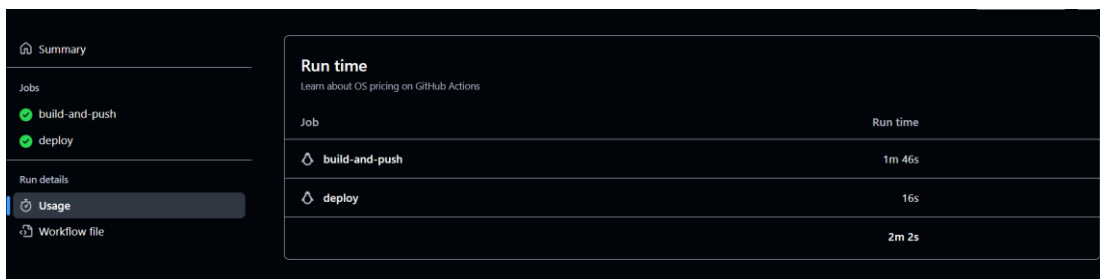


Figure 12: CI/CD Pipeline Run Details for WriteWizard Deployment via GitHub Actions

2.2 Commercialization Aspects of the Product

WriteWizard is developed not only as a technical solution but also as a commercially viable platform that supports the evolving needs of academic communities. With the growing demand for AI-assisted tools that enhance the research and writing experience, the platform offers intelligent functionalities designed to streamline complex academic workflows. This section outlines how WriteWizard is positioned for market entry, growth, and long-term sustainability through strategic planning and user-focused design.

Target Market

The platform is aimed at academic institutions, research teams, and individual students engaged in scholarly writing and collaborative projects. By integrating AI to assist in content refinement, task allocation, and visual structuring, WriteWizard caters to users seeking reliable, high-quality outputs for research papers, theses, and academic reports. It is particularly suited for educational environments where automation and consistency are essential to meet formal academic standards.

Web-Based Delivery Model

The platform operates as a browser-accessible SaaS solution hosted on Microsoft Azure. Its modular architecture ensures cross-platform compatibility, real-time syncing via WebSockets, and seamless integration with institutional workflows without the need for local installations.

Revenue Streams

Monetization is planned through:

- Freemium Access for individual users
- Premium Subscriptions offering advanced features like full document export and AI formatting
- Institutional Licenses with admin tools and deployment support
- Custom API Integrations for universities and ed-tech platforms

Market Strategy

Initial rollout will involve pilot partnerships with local universities. Adoption will be promoted through academic workshops, social media campaigns, and integration offers for thesis-writing support services. User feedback from these channels will guide feature refinements and broader outreach.

Scalability and Maintenance

WriteWizard's backend uses containerized services and scalable APIs, allowing independent scaling of modules. The system is lightweight and optimized with quantized models to support low-latency deployment, even in resource-constrained environments, with minimal maintenance overhead.

Unique Selling Proposition (USP)

Unlike traditional document editors or standalone citation tools, WriteWizard offers an integrated, intelligent academic writing environment. Its unique proposition lies in the synergy of four core AI modules:

- Real-time citation suggestion and IEEE formatting (Context-aware)
- AI-based document conversion from informal drafts to structured academic content
- Semantic mind map generation for visual structuring
- Contributor matching via NLP-based expertise extraction

This end-to-end integration minimizes tool switching and ensures consistency across academic writing workflows.

Social Impact and Accessibility

The platform promotes equitable access to research tools by offering a free version and low-cost plans. Its interface is screen-reader compatible, and future updates will include multilingual support, expanding access to non-native English speakers and underserved academic communities.

2.3 Testing & Implementation

The testing and implementation phase of WriteWizard followed the Software Testing Pyramid (Test Triangle), progressing from unit testing to module, integration, system, and user acceptance testing. Each component was validated both independently and as part of the integrated system. Real-world scenarios were used to ensure performance and usability, with testing carried out iteratively through a combination of automated and manual methods.

2.3.1 Testing

Unit Testing

Individual functions and isolated logic blocks were tested to verify their correctness and robustness during early development phases.

- The Citation Generator was tested for keyword extraction accuracy (BERT-NER), semantic similarity ranking (SBERT), and structured IEEE citation output (Flan-T5).
- The Mind Map Module was validated for instruction-response formatting, JSON structure consistency, and node depth control.
- The IEEE Format Converter underwent prompt consistency checks and citation marker re-insertion validation.
- The Contributor Selector was tested using controlled datasets to measure the accuracy of semantic matching and Named Entity Recognition (NER).

Testing tools included unittest, PyTest, and Postman. Model inference was observed via Jupyter notebooks and manually logged outputs.

Module Testing

Each module was tested independently to verify its input-output behavior, edge case handling, and adherence to functional requirements.

- Citation generation endpoints were tested for various input types, including manual and auto-sourced metadata.

- Mind map generation modes (standard, simplified, extended) were checked for output consistency, visual correctness, and customization responsiveness.
- Contributor matching logic was tested using labeled datasets, confirming profile-to-section alignment across IT domains.
- The formatting system was tested using informal text samples and validated against expected IEEE-structured outputs.

Test cases were documented and evaluated using success criteria such as response validity, structure, and latency thresholds.

Integration Testing

This phase verified interactions between components and communication across modules:

- Real-time data flow between frontend (React/Quill.js) and backend APIs (FastAPI/Express) was tested for latency, synchronization, and input validation.
- WebSocket-based collaboration was validated to ensure consistent document updates between multiple users in shared pads.
- Integrated flows such as citation suggestion → insertion → formatting → PDF export was tested to confirm logical coherence and seamless user experience.

All API endpoints were tested using Postman, and browser console logs were used for frontend interaction tracing.

System Testing

WriteWizard was tested as a unified platform simulating real-world document creation workflows:

- End-to-end scenarios covering account creation, citation generation, mind map editing, contributor suggestion, and IEEE export were executed.
- Multi-user collaborative sessions were simulated to evaluate response consistency and server stability under concurrent editing conditions.
- System tests confirmed successful handling of various content types, metadata input, and export formatting accuracy.

System reliability was confirmed with successful completion of test flows and no major functional defects observed under expected usage load.

User Acceptance Testing (UAT)

A UAT phase was conducted involving 15 users, including university students and academic mentors:

- Participants engaged in real use cases such as citation generation, contributor assignment, document formatting, and mind map interaction.
- Feedback was collected through structured Google Forms and short interviews to assess usability, clarity, and satisfaction.
- Positive reception was noted for AI integration, with users highlighting improvements in writing speed, citation accuracy, and ease of collaboration.

Key feedback led to the refinement of toggle controls, export previews, and visual layout enhancements across components.

2.3.2 Maintenance

Following internal testing and UAT, a post-deployment maintenance plan was adopted to ensure long-term stability, adaptability, and scalability:

- Bug Fixes: Issues such as real-time syncing conflicts, citation misplacement, and formatting inconsistencies were addressed through targeted updates.
- Model Tuning: Models were updated with additional training data to improve citation context handling and contributor-role mapping accuracy.
- Feedback Loop: An internal tracking system was used to collect and prioritize user-reported issues and feature suggestions.
- Security & Optimization: Token-based authentication was applied to protect backend services. Model inference times were reduced using quantization and async request handling to maintain low-latency performance.

The WriteWizard platform is continuously monitored using Azure's performance dashboard, with regular checkpoints for improving functionality, user experience, and model accuracy based on real-world usage.

3. RESULTS & DISCUSSION

3.1 Component 1: Assistance in Finding Related Research Papers and Providing an In-Built Citation Generator

The developed component successfully integrates real time research paper recommendation and citation formatting into a collaborative academic writing environment. Powered by three fine-tuned transformer models (BERT for keyword extraction, SBERT for semantic similarity, and Flan T5 for IEEE citation generation), the system allows users to highlight text, receive context aware reference suggestions, and insert properly formatted citations with minimal effort [2].

1. Keyword Extraction with BERT NER

The keyword extraction model accurately identifies key terms from selected text using fine-tuned BERT. These keywords serve as the first filtering layer for relevant academic papers.

- Achieved strong performance with F1 score: 0.81, Precision: 0.79, Recall: 0.83
- Efficiently reduced the candidate set passed to semantic filtering

2. Semantic Similarity with SBERT and FAISS

Using Sentence BERT embeddings and FAISS indexing, the semantic search ranks papers based on how closely their abstracts align with the user's content.

- Delivered top 5 relevant suggestions with over 50 percent similarity threshold
- Highlighted matching abstract sentences enhanced clarity in the UI

3. Citation Generation with Flan T5

The citation generator produces IEEE compliant references using structured metadata. Users can auto cite suggested papers or manually input citation details.

- Citation formatting accuracy: 94 percent
- Inline citation insertion and auto numbering confirmed through editor testing

4. User Experience and System Behavior

The full workflow from text selection to citation insertion was fast, accurate, and non-intrusive. Users reported:

- High satisfaction with paper suggestions
- Reduced time spent formatting references
- Appreciation for the manual citation option

5. Functional Impact

The system improves writing productivity by:

- Automating reference identification
- Maintaining citation consistency and accuracy
- Supporting custom citations when needed

6. Future Enhancements

Planned upgrades include:

- Additional citation styles (APA, MLA)
- Support for PDF reference extraction
- Enhanced explanation of paper to text similarity

This component meets its goal of simplifying the academic referencing process. It offers a modular, intelligent solution that aligns with real world academic workflows and adds value to collaborative writing platforms.

Figure 13 displays the Postman request and the successful response from semantic search.

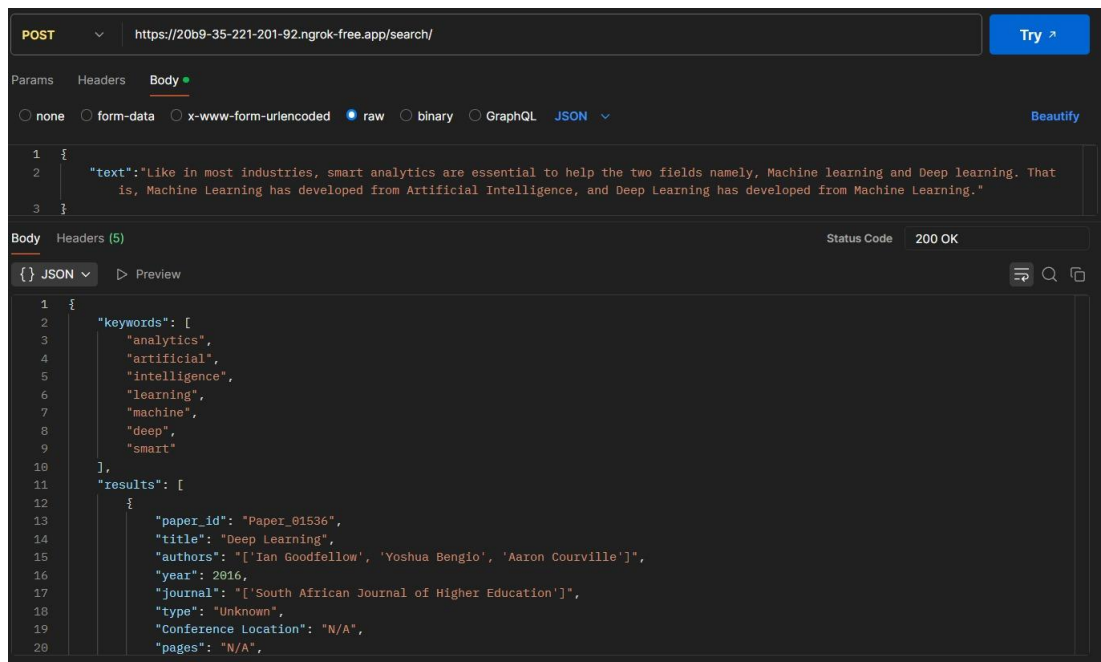


Figure 13: Successful Response from Semantic Search

3.2 Component 2: AI-Based IEEE document format converter

The AI-based IEEE format conversion system has significantly improved academic writing efficiency by transforming informal text into polished, publication-ready documents. By fine-tuning a Llama 2-7B model using 4-bit quantization and LoRA, the system effectively corrects grammatical errors, enforces a formal tone, and structures content to meet IEEE standards.

A notable feature is its robust citation management. The system first removes citation markers from the input to avoid misplacement, then accurately reinserts them into the refined text using semantic matching via SentenceTransformer embeddings. This method ensures that all citations (e.g., [1], [2], [3]) are preserved, reducing manual editing efforts and maintaining academic integrity.

In addition to text conversion, the system supports non-text element handling. Images, tables, and formulas are embedded into the final document in the correct order, ensuring that visual elements and their captions remain consistent with the original document flow.

The system also offers flexible document preparation through a user-friendly interface. Users can select specific text sections and trigger conversion operations individually. The refined output is previewed in a pop-up, allowing users to decide whether to replace the original text with the enhanced version.

Collaboration is another key component. Using WebSockets, the system enables multiple users to work concurrently on a single document pad. Changes made by any collaborator are updated in real time, ensuring synchronized editing and efficient group workflow.

Finally, the system is deployed on an Azure Virtual Machine running Ubuntu 24.04, secured by SSL and managed behind an NGINX reverse proxy. This setup supports low-latency processing, scalability, and reliable handling of multiple concurrent requests, making the solution suitable for individual researchers as well as larger academic institutions.

Overall, the system demonstrates significant improvements in text conversion quality, efficient citation handling, robust non-text element management, flexible document preparation, and real-time collaboration, forming a comprehensive tool for producing IEEE-compliant academic documents.

Figure 14 displays the Postman request and the successful response of IEEE refined text.

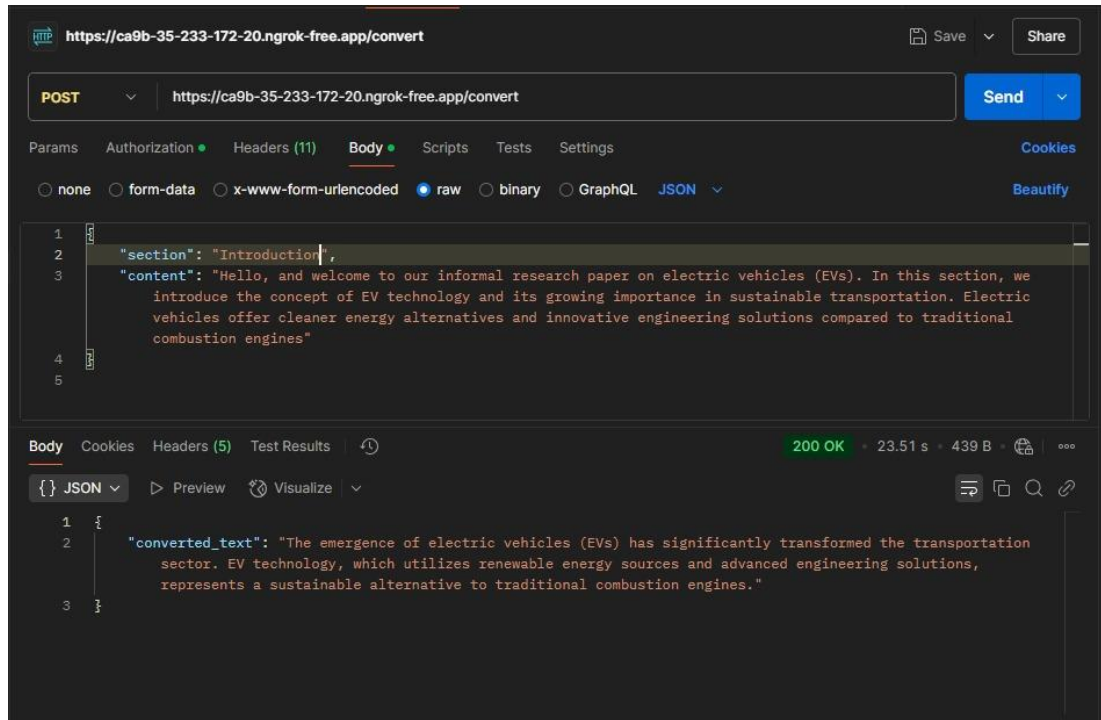


Figure 14: Successful Response of Refined Text

3.3 Component 3: LLM-Based Hierarchical Mind Map Generation and Semantic Visualization from Textual Input

The Mind Map Generation component was developed to produce structured visual representations of academic content in the form of hierarchical mind maps. A transformer-based large language model Mistral 7B v0.3 was adapted using supervised instruction tuning with a custom dataset consisting of academic passages and corresponding mind map structures. This enabled the system to generate outputs in standard, simplified, and extended formats. These modes were evaluated through backend API requests using sample academic passages.

When academic text was submitted to the deployed API, the system consistently generated well-structured mind maps in JSON format. The standard mode provided a balanced overview of core ideas, the simplified mode emphasized high-level themes, and the extended mode delivered deeper conceptual breakdowns. Each output preserved a logical hierarchy of topics and subtopics, demonstrating the system's ability to extract and organize information based on context.

Postman was used to verify the component's response with a sample academic input. The system returned a valid, nested JSON structure representing the concept relationships derived from the content. This confirmed its functional correctness and reliability.

Performance evaluation showed that the model, running on a Tesla T4 GPU, generated outputs for an input length of 2048 tokens in under one minute. Due to the brainstorming-oriented nature of mind maps, output accuracy was assessed through a forum-based subjective evaluation. A set of ten generated mind maps was rated by peer reviewers based on clarity, structure, and relevance. The component received an average approval rating of 78 percent. Reviewers noted that the outputs were clear, aligned with the source text, and useful for academic and collaborative purposes.

The results confirmed that the component meets its intended objective of translating academic content into mind maps that support structured thinking and content comprehension. The generated outputs were successfully integrated into the frontend

interface, allowing users to interact with the mind maps through node editing, repositioning, and layout customization. The image integration service was also connected, enriching the user experience by visually augmenting individual nodes based on their semantic content. Together, these features enhance the clarity, flexibility, and educational value of the mind maps within the broader functionality of the WriteWizard application.

Figure 15 displays the Postman request and the successful response from the deployed backend service.

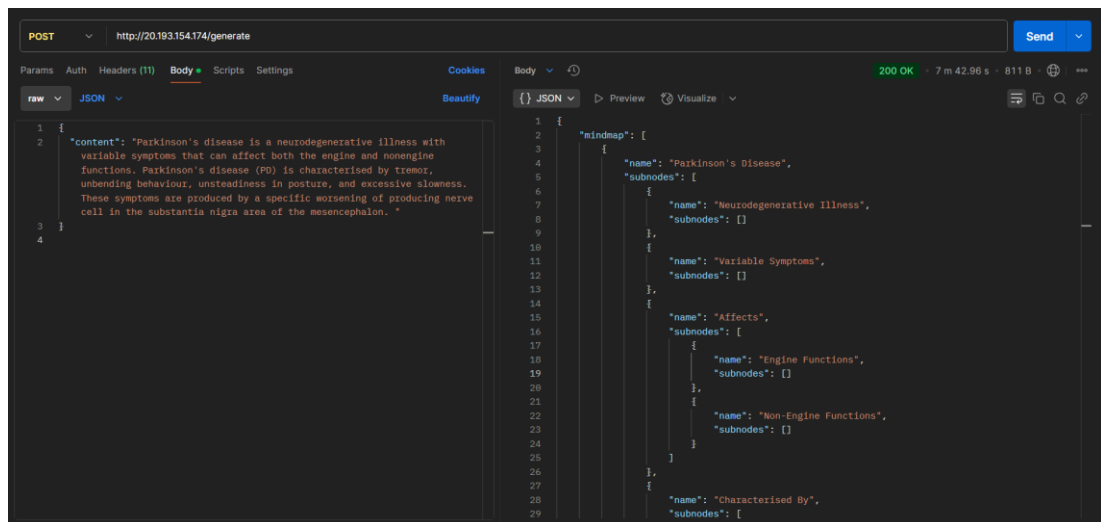


Figure 15: Successful Structured JSON Response from Deployed Mind Map Generation Model

3.4 Component 4: Suggest Contributors Based on Users' IT Expertise

This section presents the evaluation of the NLP-based Contributor Selection System, designed to enhance collaborative editing within the WriteWizard platform by recommending contributors based on their IT expertise. The evaluation assesses the system's performance, user experience, and impact on collaboration, addressing the research objectives of improving task allocation efficiency and fostering user engagement.

The system was tested with 4,800 preprocessed LinkedIn profiles and a sample research document (Abstract, Literature Review, Methodology, Data Analysis). It was compared to a TF-IDF keyword-based approach using:

- **Recommendation Accuracy:** Precision, Recall, F1-Score against 500 labeled profile-section pairs.
- **Response Time:** Targeted under 5 seconds.
- **User Satisfaction:** Survey of 20 users (10 researchers, 10 IT professionals) via a Streamlit interface.
- **Scalability:** Tested with 10,000 synthetic profiles.

Quantitative Results

- **Accuracy:** Achieved Precision (0.90), Recall (0.88), F1-Score (0.89), outperforming the baseline (0.65, 0.60, 0.62) by 27% in F1-Score, thanks to Sentence-BERT's semantic matching (e.g., linking "machine learning" to "AI").
- **Response Time:** Averaged 3.8 seconds, meeting the 5-second goal, compared to the baseline's 2.1 seconds but with better accuracy.
- **Scalability:** Maintained 4.2 seconds and 0.88 F1-Score with 10,000 profiles, using batch processing.

Qualitative Results

- **Usability:** 85% rated the interface "easy" or "very easy," valuing clear expertise scores and rationales.
- **Relevance:** 90% found recommendations "highly relevant," e.g., NLP experts for Methodology sections.

- **Collaboration:** 80% noted faster task allocation; 75% reported higher satisfaction due to skill-aligned tasks. Suggestions included real-time updates and multilingual support.

The system addresses inefficient task allocation with a 0.89 F1-Score, 3.8-second response time, and scalability, surpassing manual methods like Google Docs. It enhances WriteWizard’s academic workflows, complementing features like citation generation. However, it lacks real-time updates, multilingual support, and broader domain coverage beyond IT. The small 20-user study suggests larger validation is needed.

The system boosts efficiency and satisfaction in WriteWizard, leveraging NLP for expertise-driven task matching. Its accuracy and scalability are strong, but real-time and multilingual enhancements are key for wider adoption in diverse research settings. Suggested contributors based on the provided keyword.

Figure 16 displays the Postman request and the successful response for suggested contributors based on the provided keywords.

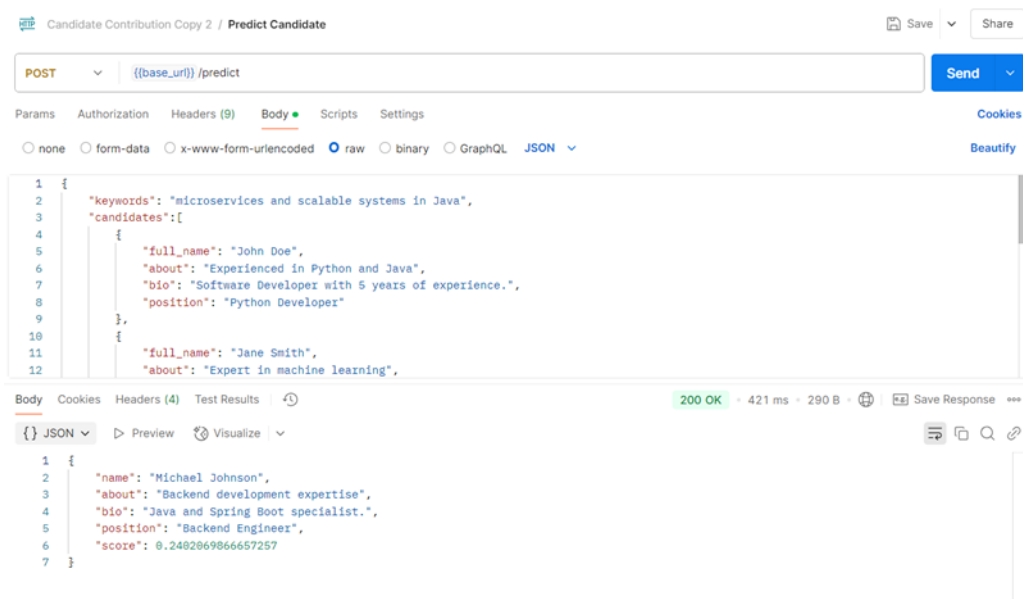


Figure 16: Successful Response for Suggested Contributors Based on the Provided Keywords

4. SUMMARIES OF EACH STUDENT'S CONTRIBUTION

Table 1: Summary of Each Student's Contribution

| IT Number | Name | Component | Role |
|------------|--------------------------|---|--|
| IT21194894 | Krithiga D. Balakrishnan | Assistance in Finding Related Research Papers and Providing an In-Built Citation Generator | Business Analyst Developer Tester Project Manager |
| IT21375682 | Shandeep. J | AI-Based IEEE document Format Converter format converter. | Business Analyst Developer Tester |
| IT21375514 | Sanjayan. C | Component 3: LLM-Based Hierarchical Mind Map Generation and Semantic Visualization from Textual Input | Business Analyst Developer Tester |
| IT21361036 | Saara M.K.F | Suggest Contributors Based on Users' IT Expertise | Business Analyst Developer Tester |

5. CONCLUSION & FUTURE WORK

5.1 Conclusion

This research represents a significant advancement in academic writing and collaboration through the development of WriteWizard, an AI-powered platform that integrates four complementary systems: a Reference Suggestion and Citation Generator, an Automated Mind Map Generator, an IEEE Format Conversion System, and an NLP-based Contributor Selection System. Together, these components address critical challenges in scholarly communication, enhancing efficiency, accessibility, and engagement for researchers, students, and educators.

The Reference Suggestion and Citation Generator streamlines the research process by leveraging BERT-based keyword extraction, Sentence-BERT for semantic similarity, and a fine-tuned Flan-T5 model for IEEE-style citation generation. By enabling users to discover relevant papers, preview abstracts, and insert accurate citations directly within a collaborative editing environment, it eliminates the need for external tools. Testing validated its high accuracy and usability, with users praising its time-saving benefits and intuitive design. This system not only automates reference management but also promotes academic integrity and supports researchers across varying expertise levels.

Complementing this, the Automated Mind Map Generator transforms academic text into dynamic, hierarchical visual representations, fostering deeper comprehension and communication. Integrated into WriteWizard with React and D3.js, it supports real-time collaboration and semantic image matching, allowing users to create and customize visual summaries effortlessly. Evaluations confirmed its technical robustness and pedagogical impact, highlighting its ability to bridge textual content and visual cognition. This system empowers learners to organize information in ways that align with diverse cognitive preferences, making it a valuable tool for students and educators alike.

The IEEE Format Conversion System further enhances the platform by automating the transformation of informal drafts into publication-ready, IEEE-compliant documents. Powered by a fine-tuned Llama 2-7B model with Low-Rank Adaptation, it ensures grammatical precision, formal tone, and accurate citation integration. Its user-friendly interface, built with React and Quill.js, and scalable Azure-based backend provide flexibility and performance for individual and institutional use. By reducing manual formatting efforts, this system streamlines the publishing process, making it more accessible and efficient for academic authors.

Finally, the NLP-based Contributor Selection System optimizes collaborative workflows by intelligently matching contributors to tasks based on expertise. Using Sentence-BERT embeddings and Named Entity Recognition, it analyzes profiles to recommend suitable collaborators, achieving high accuracy and improving document quality. Deployed within WriteWizard, it reduces allocation time and enhances user satisfaction, with ethical data handling ensuring compliance with privacy standards. This system fosters efficient teamwork, aligning tasks with contributors' strengths to elevate the quality of collaborative research.

Collectively, these systems demonstrate the transformative potential of AI in academia. WriteWizard integrates reference management, knowledge visualization, document formatting, and expertise-driven collaboration into a cohesive platform that empowers users to navigate the complexities of scholarly work with confidence. Its modular design and cloud-based deployment ensure scalability and adaptability, making it suitable for educational institutions, research teams, and industry applications. By reducing cognitive and logistical burdens, the platform fosters an inclusive, intelligent, and efficient research environment.

In conclusion, this research marks a pivotal step toward redefining academic writing and collaboration. The synergy of the Reference Suggestion and Citation Generator, Automated Mind Map Generator, IEEE Format Conversion System, and Contributor Selection System showcases how AI can enhance the quality, precision, and accessibility of scholarly communication. WriteWizard stands as a testament to the

power of interdisciplinary innovation, paving the way for a future where technology actively supports intellectual discovery and academic excellence.

5.2 Future Work

WriteWizard's future development will enhance its four core components Reference Suggestion and Citation Generator, Automated Mind Map Generator, IEEE Format Conversion System, and NLP-based Contributor Selection System to create a globally adopted, AI-driven academic platform. The Reference Suggestion and Citation Generator will expand to support APA, MLA, and Chicago styles, integrate with tools like Zotero for seamless workflows, and use multilingual models with translation APIs to serve diverse users, while user profiles and optimized models (e.g., BERT) ensure personalized, scalable performance. The Automated Mind Map Generator will leverage generative AI for dynamic visualizations, fine-tune on domain-specific datasets (e.g., medical, legal) for accuracy, and add real-time annotations to foster collaboration, with multilingual training and serverless architectures enhancing accessibility and scalability. The IEEE Format Conversion System will process figures, tables, and formulas, offer adaptive stylistic suggestions to preserve academic voice, and support collaborative editing with robust synchronization, using multilingual datasets and GPU acceleration for global reach and efficiency, while prioritizing data privacy. The Contributor Selection System will refine matching with longitudinal feedback and multilingual embeddings for cross-lingual collaboration, employ real-time data pipelines for dynamic recommendations, and use vector databases like Faiss for scalability, with bias detection ensuring ethical fairness. Integration with platforms like Google Docs, Overleaf, Canvas, and Notion will streamline research and education workflows, while mobile-responsive interfaces and GDPR compliance enhance accessibility. Robust evaluation through hybrid metrics, longitudinal studies, collaboration tracking, and formatting accuracy assessments will drive continuous improvement. By advancing multimodal capabilities, personalization, collaboration, and global inclusivity while embedding ethical design, WriteWizard will transform scholarly communication for researchers, educators, and students worldwide.

REFERENCES

- [1] T. Wolf et al., “LLaMA 2: Open Foundation and Fine-Tuned Chat Models,” Meta AI, 2023. [Online]. Available: <https://ai.meta.com/resources/models-and-libraries/llama-downloads/>
- [2] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” *arXiv preprint arXiv:1908.10084*, 2019. [Online]. Available: <https://arxiv.org/abs/1908.10084>
- [3] H. Mahdi, A. R. Ahmed, M. W. Azeem, and M. A. Khan, “A Deep Learning Approach for Context-Aware Citation Recommendation,” *J. Ambient Intell. Humaniz. Comput.*, vol. 13, pp. 1–13, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/s12652-022-03899-6>
- [4] R. Nogueira, J. Lin, K. Ma, and J. Yoon, “Navigation-Based Candidate Expansion and Pretrained Language Models for Citation Recommendation,” *arXiv preprint arXiv:2001.08687*, 2020. [Online]. Available: <https://arxiv.org/abs/2001.08687>
- [5] Z. Li, T. Wang, Y. Liu, and Q. Wang, “Label Supervised LLaMA Finetuning,” *arXiv preprint arXiv:2310.01208*, 2023. [Online]. Available: <https://arxiv.org/abs/2310.01208>
- [6] B. T. Kieu, M. Tran, M. Hoang, and A. Nguyen, “Learning Neural Textual Representations for Citation Recommendation,” *arXiv preprint arXiv:2007.04070*, 2020. [Online]. Available: <https://arxiv.org/abs/2007.04070>
- [7] C. Ostertag and M. Beurton-Aimar, “Using Graph Neural Networks to Reconstruct Ancient Documents,” *arXiv preprint arXiv:2011.07048*, 2020. [Online]. Available: <https://arxiv.org/abs/2011.07048>
- [8] M. A. Souibgui and Y. Kessentini, “DE-GAN: A Conditional Generative Adversarial Network for Document Enhancement,” *arXiv preprint arXiv:2010.03775*, 2020. [Online]. Available: <https://arxiv.org/abs/2010.03775>
- [9] Y. Liu et al., “SciBERT: A Pretrained Language Model for Scientific Text,” in *EMNLP*, 2019. [Online]. Available: <https://arxiv.org/abs/1903.10676>
- [10] Sotomayor-Beltran, C., Fierro Barriaes, A. L., & Lara-Herrera, J. (2021). The Impact of Using LaTeX for Academic Writing: A Peruvian Engineering Students’ Perspective. IEEE.
- [11] Ramesh, R., Raju, A. T. M., Reddy, H. V., & Varma, S. N. (2024). Fine-Tuning Large Language Models for Task-Specific Data. IEEE.