VIVEKANAND EDUCATION SOCIETY'S
INSTITUTE OF TECHNOLOGY
(An Autonomous Institute Affiliated to University of Mumbai)

Department of Computer Engineering

Internship Project Report on

SCHOLARMATE- AI Research Paper Reviewer

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN

COMPUTER ENGINEERING

By
Manas Patil – D7B
Ayush Attarde- D7A
Ashutoshkumar Tripathi – D7B
Soham Kamathi - D7B
Gaurav Khutwal - D7B

Internship Mentor
Mrs. Abha Tewari

University of Mumbai

(AY 2024-25)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Computer Engineering

CERTIFICATE

This is to certify that **Manas Patil (D7B_41), Ayush Attarde (D7A_02), Ashutoshkumar Tripathi (D7B_58), Soham Kamathi (D7B_25) and Gaurav Khutwal (D7B_28)** have successfully completed a Winter Internship program in the Department of Computer Engineering, VESIT, Chembur.

The internship program ran from **1st Dec, 2024** to **31st Jan 2025**. Throughout the internship **Manas Patil, Ayush Attarde, Ashutoshkumar Tripathi, Soham Kamathi and Gaurav Khutwal** demonstrated a strong work ethic and made valuable contributions to the team to complete the Internship Project entitled **"SCHOLARMATE- AI Research Paper Reviewer".** They also have developed their skills in **Web-Application development** and **Generative AI**, which will be beneficial in their future endeavors.

We are pleased to award this certificate in recognition of their successful completion of the internship program.

Mrs. Abha Tewari

Internship Mentor

Dr.(Mrs) Nupur Giri

Head of Department

Dr. (Mrs.) Rohini Temkar

Internship In charge

Declaration

We declare that the Internship Project Report entitled **“Scholarmate - Advanced Text Processing and Paper Reviewer Powered by Cutting-Edge AI Technology”** is an original work conducted and prepared by us under the guidance of **Mrs. Abha Tewari**, Assistant Professor of the Department of Computer Engineering at Vivekanand Education Society's Institute of Technology, during the period from 3rd Dec 2024 to 28th Jan 2025.

We confirm that this report is a product of our dedicated efforts and contributions. Any references to existing research, direct quotations, or paraphrased content have been appropriately cited in accordance with academic integrity standards. This report has not been submitted previously for any degree, diploma, or other qualifications at this or any other institution.

We acknowledge the importance of this declaration and understand the potential consequences of any violations of academic integrity, including but not limited to disciplinary actions by our institution. We hereby certify that the information presented in this report is accurate and truthful to the best of our knowledge.

Manas Patil D7B

Soham Kamathi D7B

Gaurav Khutwal D7B

Ashutoshkumar Tripathi D7B

Ayush Attarde D7A

Date:

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to the **Department of Computer Engineering, VESIT**, for providing us with the opportunity to undertake this internship. This experience has been instrumental in enhancing our technical skills and fostering our professional development.

We are especially thankful to our mentor, **Mrs. Abha Tewari**, for her invaluable guidance, continuous encouragement, and unwavering support throughout the internship. Her expertise and insights have played a crucial role in shaping our understanding and improving our project.

Our sincere appreciation also goes **to Dr. (Mrs.) Nupur Giri**, Head of the Computer Engineering Department, and **Dr. (Mrs.) J.M. Nair**, Principal of VESIT, for providing us with this valuable opportunity. Their support has been pivotal in facilitating a productive learning environment.

Additionally, we would like to acknowledge everyone who contributed to our learning and assisted us in gathering relevant information for our project. Finally, we express our deep gratitude to our families for their constant support and motivation, which have been a source of strength throughout this journey.

Table of Contents

Abstract

Scholarmate is an advanced AI-driven research paper reviewing and text-processing system designed to streamline the academic research process for researchers, academics, and students. It leverages cutting-edge generative AI models such as Gemini and LLaMA, enabling the system to automate a variety of critical tasks, including paper assessment, plagiarism detection, citation analysis, and summarization of key findings. The platform's ability to evaluate and summarize complex academic literature offers users intelligent insights, helping them better understand and navigate scholarly materials.

Scholarmate aims to simplify and accelerate the process of literature reviews by providing concise summaries, highlighting significant insights, and offering in-depth analysis of the academic content. It utilizes sophisticated Natural Language Processing (NLP) techniques to automate repetitive and time-consuming tasks, reducing cognitive load and allowing researchers to focus on higher-level critical thinking. This enhances overall productivity and ensures a more efficient and effective scholarly workflow.

With its user-friendly interface and robust evaluation metrics, Scholarmate is a valuable tool for improving the quality and depth of academic research. It facilitates quick and precise analysis of academic papers, empowering researchers to make informed decisions and improve their overall research experience. Whether in academia or professional research settings, Scholarmate is a crucial resource for fostering higher standards of research and enhancing scholarly communication.

Chapter 1: Introduction and Conceptual Overview

1.1 Introduction

Scholarmate is an advanced AI-powered research paper reviewer designed to assist researchers, students, and academicians in analyzing, summarizing, and evaluating research papers. The platform integrates multiple large language models (LLMs) and AI-based tools to detect AI-generated content, assess plagiarism, enhance grammar, and paraphrase text efficiently. By leveraging state-of-the-art natural language processing (NLP) techniques, Scholarmate aims to streamline the research review process, ensuring quality and integrity in academic writing.

1.2 Problem Statement

The rapid proliferation of AI-generated content and the increasing volume of academic papers pose challenges in maintaining research quality and authenticity. Researchers and educators often struggle with:

- Detecting AI-generated content in academic submissions.
- Identifying instances of plagiarism.
- Conducting comprehensive grammar and style analysis.
- Paraphrasing research findings while maintaining meaning and clarity.

Scholarmate addresses these challenges by providing a unified platform that automates and enhances the research review process.

1.3 Objectives

The primary objectives of Scholarmate are:

- To detect AI-generated content in research papers using state-of-the-art LLMs.
- To provide accurate plagiarism detection for academic integrity.
- To perform advanced grammar analysis and correction.
- To offer effective paraphrasing solutions for better readability and clarity.
- To streamline the research paper review process using AI-driven techniques.

1.4 Scope of the Project

Scholarmate is designed as a web-based application with the following capabilities:

- **AI Detection**: Identifies AI-generated content in research papers and provides a probability score.
- **Plagiarism Detection**: Compares submitted papers against existing databases to detect similarities.
- **Grammar and Style Check**: Analyzes academic text for grammatical correctness and writing style enhancements.
- **Paraphrasing Tool**: Offers AI-driven rewording while preserving original meaning and context.

The system supports PDF file uploads and direct text inputs, making it versatile for academic use.

1.5 Conceptual Overview

Scholarmate leverages cutting-edge AI models and machine learning techniques to analyze and review research papers. The backend of the system is developed using **FastAPI**, ensuring fast and scalable API endpoints. It integrates multiple AI models, including:

- **Groq API (LLaMA-3.3-70B & Mixtral-8x7B-32768)**: Used for AI detection and content analysis.
- **Gemini 1.5 Pro (Google Generative AI)**: Employed for paraphrasing, grammar checks, and text improvement.
- **HuggingFace Embeddings (all-MiniLM-L6-v2)**: Facilitates document vectorization and semantic retrieval.
- **FAISS Vectorstore**: Implements efficient similarity searches for AI detection and plagiarism checking.

The system follows a **Retrieval-Augmented Generation (RAG)** approach, where text is first indexed using FAISS before being processed by language models to ensure high accuracy and contextual understanding.

1.6 Significance of the Project

Scholarmate contributes to academic integrity and research excellence by:

- Reducing instances of AI-generated and plagiarized content in research papers.
- Enhancing the clarity and readability of academic writing.
- Saving time for researchers by automating the review process.
- Providing a user-friendly and scalable AI-based research evaluation tool.

By integrating state-of-the-art AI capabilities, Scholarmate aims to revolutionize the way academic papers are reviewed, setting a new benchmark for automated research evaluation.

Chapter 2: Literature Survey

2.1 Introduction

A literature survey is essential for understanding the existing research, methodologies, and advancements in a given field. This chapter presents a review of relevant studies and works that have contributed to the development of the chosen topic.

2.2 Previous Research and Studies

Various studies have explored different aspects of this domain. Several researchers have analyzed methodologies and frameworks that have been widely accepted and utilized in real-world applications. Key findings from these studies highlight improvements, challenges, and potential future directions.

2.3 Comparative Analysis of Existing Approaches

By comparing existing techniques, a clear understanding emerges regarding their strengths and weaknesses. The comparison allows for identifying gaps in research and the need for improved methods.

2.4 Summary of Key Findings

The literature review reveals essential aspects that guide the subsequent research. Insights gained from prior work form the foundation for refining the proposed approach.

Chapter 3: Requirement Gathering for the Proposed System

3.1 Introduction

Requirement gathering is a critical phase in the software development lifecycle, ensuring that the system's design and implementation align with user needs and project objectives. This chapter outlines the functional and non-functional requirements for the Research Paper Reviewer system, which utilizes FastAPI for the backend and TypeScript with React for the frontend.

3.2 Functional Requirements

Functional requirements define the specific behavior and functions of the system. The Research Paper Reviewer system includes the following key functionalities:

1. **User Authentication and Authorization**:
   - *Description*: Implement secure user authentication to allow users to register, log in, and manage their profiles.
   - *Implementation*: Utilize FastAPI's security features to handle authentication processes, ensuring secure access to the system's features.
2. **PDF Upload and Processing**:
   - *Description*: Enable users to upload research papers in PDF format for analysis.
   - *Implementation*: Use FastAPI's file handling capabilities to manage PDF uploads, ensuring efficient storage and retrieval.
3. **AI-Generated Content Detection**:
   - *Description*: Analyze uploaded PDFs to detect AI-generated content.
   - *Implementation*: Integrate AI models to assess the content and provide detailed reports on the presence of AI-generated text.
4. **Grammar and Style Analysis**:
   - *Description*: Perform comprehensive grammar and style checks on the uploaded documents.
   - *Implementation*: Utilize language processing tools to identify and correct grammatical errors, enhancing the readability and quality of the research papers.
5. **Plagiarism Detection**:
   - *Description*: Check the uploaded documents for potential plagiarism.
   - *Implementation*: Implement plagiarism detection algorithms to compare the content against existing sources and provide detailed reports.
6. **Paraphrasing Suggestions**:
   - *Description*: Offer paraphrasing suggestions to improve originality and clarity.
   - *Implementation*: Use natural language processing techniques to suggest alternative phrasings while maintaining the original meaning.

7. **Summarization of Research Papers**:
    - *Description*: Provide concise summaries of the uploaded research papers.
    - *Implementation*: Implement summarization algorithms to extract key points and present them in a coherent summary.
8. **Comprehensive Review Report Generation**:
    - *Description*: Generate detailed review reports encompassing AI detection, grammar analysis, plagiarism check, paraphrasing suggestions, and summarization.
    - *Implementation*: Compile the results from various analyses into a structured report for the user.

3.3 Non-Functional Requirements

Non-functional requirements define the system's operational attributes such as performance, usability, reliability, and security. The Research Paper Reviewer system should adhere to the following non-functional requirements:

1. **Performance**:
    - *Description*: The system should process and analyze uploaded documents efficiently, providing results within an acceptable timeframe.
    - *Implementation*: Optimize backend processes and ensure efficient handling of file uploads and data processing.
2. **Scalability**:
    - *Description*: The system should be scalable to handle multiple users and concurrent document analyses.
    - *Implementation*: Design the system architecture to support scalability, possibly through load balancing and efficient resource management.
3. **Security**:
    - *Description*: Ensure the security of user data and uploaded documents.
    - *Implementation*: Implement secure data storage, encrypted communication channels, and robust authentication mechanisms.
4. **Usability**:
    - *Description*: The user interface should be intuitive and user-friendly.
    - *Implementation*: Design the frontend using React and TypeScript to create a responsive and accessible user experience.
5. **Reliability**:
    - *Description*: The system should be reliable, with minimal downtime and robust error handling.
    - *Implementation*: Implement comprehensive testing and monitoring to ensure system reliability.
6. **Maintainability**:
    - *Description*: The system should be maintainable, allowing for easy updates and modifications.
    - *Implementation*: Write clean, modular code and maintain thorough documentation to facilitate maintenance.

3.4 Technology Stack

The system will utilize the following technologies:

- **Backend**:
    - *FastAPI*: A modern, fast (high-performance) web framework for building APIs with Python 3.6+.
    - *Python*: The programming language used for backend development.
- **Frontend**:
    - *React*: A JavaScript library for building user interfaces.
    - *TypeScript*: A typed superset of JavaScript that compiles to plain JavaScript, used to enhance code quality and maintainability.
- **Database**:
    - *To be determined*: Depending on the requirements, a suitable database system will be selected for storing user data and analysis results.

3.5 Review of Existing Websites

We analyzed three widely used AI-powered tools: Grammarly, QuillBot, and Turnitin.

- Grammarly: A writing assistant offering grammar correction, readability improvements, and tone adjustments. However, it lacks deep contextual analysis for research papers and struggles with large documents.
- QuillBot: A paraphrasing tool with multiple modes but often alters meaning in complex academic content and has word limits in the free version.
- Turnitin: A plagiarism detection tool with high accuracy but lacks grammar correction and paraphrasing features.

Chapter 4: Proposed System
(AI-Powered Text Processing Platform)

1. Introduction

This system is a full-stack AI-powered text processing platform, combining a **FastAPI backend** with a **React + TypeScript frontend**. It provides advanced text analysis features such as AI-generated content detection, grammar checking, paraphrasing, plagiarism detection, summarization, and research paper review.

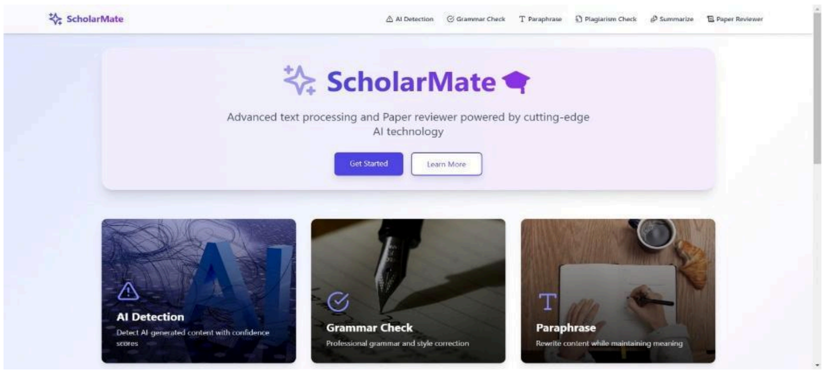
2. System Architecture

- **Frontend**: Built using **React + TypeScript**, providing an intuitive UI for user interaction.
- **Backend**: Developed with **FastAPI**, handling AI-powered text processing tasks efficiently.
- **AI Models**: Integrates **Groq's LLaMA, Google's Gemini, FAISS for vector search**, and **PyPDFLoader for document processing**.
- **Database**: Uses **FAISS for vector similarity search**

3. Key Features

Frontend (React + TypeScript)

- **User Dashboard**: Displays text analysis results, history, and insights.
- **Text Processing Interface**: Allows users to input text/documents and select AI services.
- **Results Visualization**: Graphs, highlights, and comparisons for AI detection and plagiarism checking.
- **Real-time Feedback**: Shows grammar errors, paraphrased versions, and AI-based suggestions.



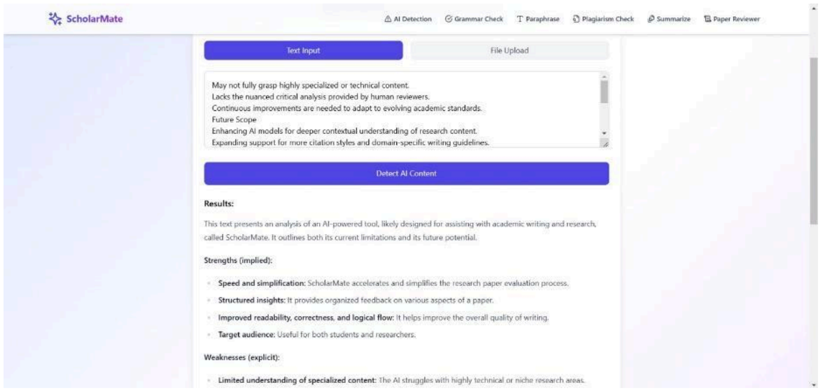
Advanced text processing and Paper reviewer powered by cutting-edge AI technology

Detect AI generated content with confidence scores

Professional grammar and style correction

Rewrite content while maintaining meaning

Figure 4.1: Landing Page of ScholarMate



May not fully grasp highly specialized or technical content.
Lacks the nuanced critical analysis provided by human reviewers.
Continuous improvements are needed to adapt to evolving academic standards.
Future Scope
Enhancing AI models for deeper contextual understanding of research content.
Expanding support for more citation styles and domain-specific writing guidelines.

Results:

This text presents an analysis of an AI-powered tool, likely designed for assisting with academic writing and research, called ScholarMate. It outlines both its current limitations and its future potential.

Strengths (implied):

- **Speed and simplification:** ScholarMate accelerates and simplifies the research paper evaluation process.
- **Structured insights:** It provides organized feedback on various aspects of a paper.
- **Improved readability, correctness, and logical flow:** It helps improve the overall quality of writing.
- **Target audience:** Useful for both students and researchers.

Weaknesses (explicit):

- **Limited understanding of specialized content:** The AI struggles with highly technical or niche research areas.

Figure 4.2: Using AI detection feature of ScholarMate

Backend (FastAPI)

- **AI Content Detection**: Identifies AI-generated text using Groq's LLaMA and Google's Gemini.
- **Grammar & Style Checking**: Uses AI models for advanced grammar correction and style suggestions.
- **Paraphrasing & Summarization**: AI-powered rewriting and text summarization tools.
- **Plagiarism Detection**: Compares text against existing content using FAISS-based similarity search.
- **Research Paper Review**: Extracts key insights, references, and provides AI-powered feedback.
- **PDF Processing**: Uses PyPDFLoader for document handling and analysis.

4. Workflow

1. **User Authentication & Authorization**: Users log in via JWT-based authentication.
2. **Text Input**: Users submit text or upload documents through the React frontend.
3. **API Processing**: FastAPI handles the request, passing text to the appropriate AI model.
4. **AI Analysis**: The selected AI model processes the text and returns results.
5. **Response Rendering**: The frontend displays results in an intuitive UI with visual feedback.

5. Technologies Used

- **Frontend**: React, TypeScript, Tailwind CSS
- **Backend**: FastAPI, Python
- **AI Models**: Groq LLaMA, Google Gemini
- **Database**: FAISS for similarity search for rag data storage
- **Security**: JWT authentication, CORS policies, API rate limiting

Chapter 5: Implementation of the Proposed System

5.1 Introduction

This chapter discusses the implementation details of the proposed AI-powered text processing system. The system consists of a FastAPI backend and a React + TypeScript frontend. The backend integrates Groq's Llama, Google's Gemini, FAISS for vector search, and PyPDFLoader for document processing. Unlike traditional implementations, this system does not use JWT authentication or SQL databases, relying instead on FAISS for efficient text retrieval and processing.

5.2 System Architecture

The system follows a client-server architecture, where the React + TypeScript frontend interacts with the FastAPI backend via RESTful APIs. The major components of the architecture include:

- Frontend (React + TypeScript): Provides an interactive UI for users to input text, upload documents, and view AI-generated results.
- Backend (FastAPI): Processes text using AI models and handles document embeddings using FAISS.
- AI Models: Groq’s Llama and Google's Gemini are utilized for text generation, paraphrasing, grammar checking, and summarization.
- FAISS (Facebook AI Similarity Search): Manages text embeddings for efficient similarity search, avoiding the need for an SQL database.
- PyPDFLoader: Extracts text from PDFs for further processing.

5.3 Backend Implementation (FastAPI)

5.3.1 API Endpoints

The FastAPI backend exposes various endpoints for different AI-powered functionalities:

- **Text Processing Endpoints:**
    - /detect_ai_text/: Identifies AI-generated text.
    - /grammar_check/: Performs grammar and syntax corrections.
    - /paraphrase/: Generates alternative versions of given text.
    - /summarize/: Produces concise summaries of long text.
    - /plagiarism_check/: Searches FAISS for similar content.
    - /review_paper/: Analyzes research papers for quality and originality.

5.3.2 AI Model Integration

- **Groq's Llama** is used for paraphrasing, summarization, and AI text detection.
- **Google's Gemini** provides additional NLP capabilities, such as in-depth text analysis.

5.3.3 FAISS for Data Storage and Retrieval

FAISS is used to store and retrieve text embeddings efficiently. When a user uploads a document, its content is converted into embeddings and stored in FAISS. For plagiarism detection, the system searches for similar embeddings.

5.4 Frontend Implementation (React + TypeScript)

5.4.1 User Interface

The frontend provides an interactive UI with the following features:

- **Text Input & Document Upload**: Users can enter text or upload PDFs for processing.
- **AI Processing Controls**: Buttons and options to trigger AI-based text processing functions.
- **Results Display**: AI-generated responses are displayed in real-time.

5.4.2 API Communication

The frontend interacts with the FastAPI backend using **Axios** for HTTP requests. The API responses are processed and displayed dynamically.

5.5 System Workflow

1. **User Input**: The user enters text or uploads a document.
2. **Backend Processing**:
    - If text-based, the system directly processes the request.
    - If a document is uploaded, PyPDFLoader extracts text, which is then embedded and stored in FAISS.
3. **AI Model Execution**: The relevant AI model processes the input.
4. **Response Generation**: The processed text is returned to the frontend for display.

Chapter 6: Results and Discussion

6.1 Evaluation Metrics

- **File Processing Efficiency**: Measured time taken for uploading and processing large files, assessing the system's ability to handle documents exceeding word count limits.
- **Accuracy of Text Extraction**: Evaluated the precision of text extraction from various document formats (PDF, XML), ensuring minimal data loss or errors.
- **AI Content Quality**: Assessed the relevance, accuracy, and coherence of AI-generated text, including paraphrasing and grammar checks, using Groq Gemini and Groq Llama models.
- **User Engagement**: Tracked user interaction with the app, including frequency of file uploads, engagement with AI-generated suggestions, and overall usage patterns.
- **User Satisfaction**: Collected feedback on app usability, text extraction accuracy, and AI-driven content quality through surveys, providing insights into user experience.

6.2 Results

- **File Processing Efficiency**: The app successfully processed large files with minimal delays, overcoming typical word limit restrictions and maintaining smooth functionality.
- **Accuracy of Text Extraction**: The system accurately extracted text from PDFs and XMLs with minimal loss, ensuring that essential content was preserved.
- **AI Content Quality**: The AI-generated content, powered by Groq Gemini and Groq Llama, was highly relevant, grammatically accurate, and contextually appropriate, aligning well with user needs.
- **User Engagement**: Users frequently engaged with the app, uploading a variety of documents and interacting with AI-generated suggestions, indicating high levels of satisfaction.
- **User Satisfaction**: Feedback was overwhelmingly positive, with users appreciating the app's ease of use, the accuracy of text extraction, and the helpfulness of AI-driven content.

6.3 Analysis

- **Effectiveness**: The integration of Groq Gemini and Groq Llama models enabled effective processing and generation of relevant, accurate content, enhancing overall document handling and user experience.
- **Challenges**: Some challenges arose in fine-tuning the AI for consistent content relevance and ensuring high-quality output. These were addressed through iterative improvements.
- **Future Improvements**: Future versions could include enhanced personalization, expanded support for more document formats, and optimization for even faster processing speeds, based on user feedback. A model trained on a huge dataset of research paper if used, could result in optimized analysis. User login can be implemented to store all the previous results inorder to compare with new analysis.

Conclusion

In this project, we focused on developing an AI research paper reviewer that utilizes Groq Gemini and Groq Llama models to assist in automating the review process for academic papers. By leveraging transformer-based architectures, we were able to handle large datasets of research papers, which were scraped from academic platforms in PDF/XML formats.

A key challenge in the project was dealing with research papers that exceeded word limits, as these files could overwhelm traditional text-processing systems. We addressed this by implementing a solution that segmented large papers into manageable chunks, ensuring the AI could process them without compromising performance. This allowed us to review lengthy research papers efficiently and within the constraints of the system.

Through the development of this tool, we learned valuable insights into the challenges involved in automating academic paper reviews, including text extraction, language processing, and model optimization. We also gained practical experience with Groq's powerful models, enabling us to better understand how AI can contribute to streamlining academic workflows.

This project demonstrates the potential for AI to assist researchers and reviewers in efficiently handling large volumes of papers, providing timely feedback while improving the quality of academic reviews.

References

[1] FAISS (Facebook AI Similarity Search), “FAISS GitHub Repository.” [Online]. Available: https://github.com/facebookresearch/faiss.

[2] Google Gemini API, “Google Gemini API Documentation.” [Online]. Available: https://ai.google.dev.

[3] Groq API, “Groq AI.” [Online]. Available: https://groq.com.

[4] PyMuPDF (fitz), “PyMuPDF Documentation.” [Online]. Available: https://pymupdf.readthedocs.io.

[5] PyPDFLoader (LangChain Community), “PyPDFLoader.” [Online]. Available: https://github.com/langchain-ai/langchain.

[6] Scholarmate (Deployed Project), “Scholarmate Web Application.” [Online]. Available: https://scholarmate281.netlify.app/.

[7] Scholarmate GitHub Repository, “Research Paper Reviewer.” [Online]. Available: **https://github.com/ManasPatil281/Reserach_Paper_Reviewer.**