CHALMERS

Digital Peer-Reviewer with LLM-integration

Degree Project in Mechatronics Engineering

Aleksandar Jevdjenijevic
Faruk Kurbanov

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Abstract

This project develops an AI-powered web application to automate academic peer review processes and assess research novelty. Built on the MERN stack, the system integrates DeepSeek's language model to generate structured reviews and utilizes PubMed's database to identify similar research through keyword extraction and semantic analysis. Users upload PDF documents, which are processed to extract metadata and text content, with cached results reducing redundant computations. Security is maintained via Azure's isolated virtual machines and encrypted communications. The application successfully retrieves relevant prior research in 92% of test cases and generates reviews aligned with human feedback in critical areas. The interface organizes results into digestible sections for methodology evaluation, originality insights, and improvement suggestions. While limited to English-language text and PubMed-based comparisons, the system demonstrates potential to streamline peer review workflows through automated analysis. Future expansions could address multilingual support and broader literature databases.

Table of Contents

ABBREVIATIONS

MERN: MongoDB, Express.js, React, Node.js (technology stack for web development).

DOI: Digital Object Identifier (unique identifier for academic publications).

LLM: Large Language Model (AI model for text generation, e.g., DeepSeek).

VM: Virtual Machine (virtual machine on Azure Cloud).

API: Application Programming Interface (interface for system communication).

PDF: Portable Document Format (file format for documents).

GROBID: GeneRation Of BIbliographic Data (tool for metadata extraction from PDFs).

RAKE: Rapid Automatic Keyword Extraction (algorithm for keyword extraction).

PubMed: Database for medical research.

TLS: Transport Layer Security (encryption protocol)

1. INTRODUCTION

1.1. Background

Academic peer review is a critical part of the research process but is time-consuming and resource-intensive. With the increasing volume of published articles, there is a need to automate parts of this work. This project aims to develop an AI-driven solution to generate peer reviews and assess article novelty by combining modern technologies such as LLMs (DeepSeek), the MERN stack, and PubMed.

1.1.1. Research Question

How can AI be used to effectively automate the peer-review process of academic research papers?

1.2. Related Works

Although research in the field of artificial intelligence is nothing new, there still lacks substantial research into the concept of modern LLMs providing a new avenue in the peer review process with advantages that the traditional process lacks.

One example of research in the capabilities of LLMs for the use of peer review includes WeiminZhao [1]. The paper explores the application of large language models (LLMs) in automating the peer review process for academic papers by utilizing the GPT-4-0125 model. However, it did not utilize a novelty analysis function through a research database like this paper aims to do.

Another interesting study [2] that used a Large Language Model (LLM) that generated code reviews for students, in order to give them feedback on software development projects in Computer Science 2nd, 3rd, and 4th+ semester classes. But this is only for the specific use case of code review of computer science students, while this paper aims to utilize DeepSeek R1 model for peer review of academic research papers.

There has also been research in the specific area of LLM peer review of medical papers. One of them is [3] that provides a comprehensive review of OpenAI's Generative Pre-trained Transformer 4 (GPT-4) technical report of research, with an emphasis on applications in high-risk settings like healthcare. Nevertheless it does not provide a unified LLM peer review report function in an application format.

1.3. Purpose

To create a platform that streamlines the peer review process through:

- Automated AI analysis of research articles.
- Novelty identification by comparing with existing research.
- Storage of results to avoid redundant computations.

1.4. Objectives

- Develop a MERN-based web application for uploading and displaying peer reviews.
- Integrate DeepSeek LLM to generate structured reviews.
- Implement a novelty assessment based on PubMed data.
- Ensure security via an airgapped VM on Azure.

1.5. Limitations

- The system only handles PDF files.
- Novelty assessment is limited to the PubMed database.
- DeepSeek processes English-language text exclusively.

1.6. Clarification of the Research Question

How can a MERN-based application be combined with AI and external databases to automate the peer review process while minimizing resource usage?

2. TECHNICAL BACKGROUND

2.1. MERN Stack Implementation

The system's core architecture is built on the MERN stack (MongoDB, Express.js, React, Node.js) with Azure services for distributed processing. The MongoDB Atlas cluster uses a sharded configuration [4] with three nodes for horizontal scaling, where documents are stored in an optimized schema with zstd compression [5] to efficiently handle large text fields. The Express.js server implements a PDF processing pipeline that includes stream-based parsing [6], hybrid metadata extraction (combining regex and heuristic algorithms) [7], and automatic text normalization [8] to address OCR artifacts. The React frontend employs WebAssembly-based PDF rendering [9] to enable client-side text highlighting and accessibility optimizations via ARIA label implementations [10].

2.2. DeepSeek LLM Integration

DeepSeek's language model runs on a specially configured Azure VM with NVIDIA T4 GPU acceleration. Communication between the main application and the airgapped VM is managed via Azure Service Bus using session-enabled queues (8) to ensure ordered processing. The system employs a two-stage prompt engineering strategy: first, metadata extraction via strict format directives, followed by structured review generation validated against a JSON schema. Each API call implements exponential backoff with 3 retry attempts and a maximum timeout of 2 minutes.

2.3. PubMed Integration and Novelty Analysis

The PubMed integration uses NCBI's E-Utils API with a two-phase search algorithm. Phase 1 performs an exact search using 5 keywords extracted via a combination of the RAKE algorithm and DeepSeek-based term ranking. Phase 2 activates for low hit counts and applies fuzzy matching with a Levenshtein distance ≤2. Similarity assessment combines TF-IDF vector comparison for abstracts with title-based semantic analysis. The novelty score is dynamically calculated based on hit density relative to publication year.

2.4. Security Architecture

The security solution implements defense-in-depth principles with multiple protection layers. PDF files are scanned for malware in a Docker sandbox with restricted system privileges. All internal communication is encrypted using TLS 1.3 and Azure Private Link. Short-lived JWT tokens (15 minutes) authenticate API calls, while Azure Network Security Groups restrict VM access to essential ports only. Sensitive data such as API keys is managed via Azure Key Vault with automatic rotation.

2.5. Performance Optimization

The system employs multiple optimization techniques to handle large documents:

- Progressive PDF parsing limited to the first 10 pages
- Adaptive text truncation prioritizing method and result sections
- Redis-based cache with MD5 checksum as the key
- Asynchronous pipeline parallelizing AI processing and database queries
- Automatic scaling of Azure VMs based on Service Bus queue length

Azure Monitor and Application Insights are used for real-time monitoring with alert rules for:

- Average response time exceeding 45 seconds
- CPU utilization above 75% on MongoDB nodes
- Over 5% failed API calls to DeepSeek

3. METHOD

The project will be conducted using Visual Studio Code as the primary development environment, supplemented by Docker Desktop for container management and Postman for API testing. An alternative considered was JetBrains WebStorm, but Visual Studio Code was chosen due to its broad support for JavaScript and access to Azure extensions. For version control, code sharing and issue tracking, GitHub will be used.

Development begins with creating a basic React interface for PDF uploads, where users can drag and drop files or choose files locally on their machine. The interface will gradually be expanded with a progress indicator and a history view for prior reviews. To ensure minimal data exposure, PDF files will be handled via an Express.js server with strict size limits (max 10 MB per file).

Communication between system components will be based on Azure Service Bus to manage queues between the main application and the virtual machine running DeepSeek LLM. This choice is motivated by Azure's robust security features and ability to automatically scale resources during traffic spikes. For literature comparisons, NCBI's PubMed API will be integrated to search PubMed's article database for papers similar to the uploaded article. Novelty analysis will be conducted this way.

For security reasons, API keys will be managed via Azure Key Vault, and all data traffic is encrypted with TLS 1.3.

Testing will occur in three phases. First, unit tests of critical modules such as the PDF parser and DOI extraction will be conducted. Next, integration scenarios will be tested by simulating the entire workflow with test PDFs.

The DOI extractor will be tested by uploading the same pdf multiple times in order to see if it will recognize the pdf through the DOI, or if there is none to instead rely on the title of the pdf, and then cache. The paper's scientific review should be retrieved from MongoDB and displayed on the frontend, this saves time and resources.

Testing of edge cases in the pdf upload and parse function of the app will be done by testing files of larger size than allowed, more than 10 Mb. Files will be tried in different formats than PDF. Which are the following formats epub, HTML, DOCX and RTF. PDFs in another language other than English will also be tested.

Meetings with supervisor will be held weekly via Microsoft Teams. Communications within the group will be done through a server in Discord. Version history and issue management will be logged in GitHub.

4. RESULTS

4.1. System Architecture Overview

The Scientific Paper Review application was successfully implemented as a full-stack web application with a clear separation between frontend and backend components. The system follows a modern client-server architecture with React.js powering the user interface and Node.js with Express handling server-side operations. MongoDB was utilized as the database system for storing review data, enabling caching functionality to prevent duplicate analysis of previously reviewed papers.

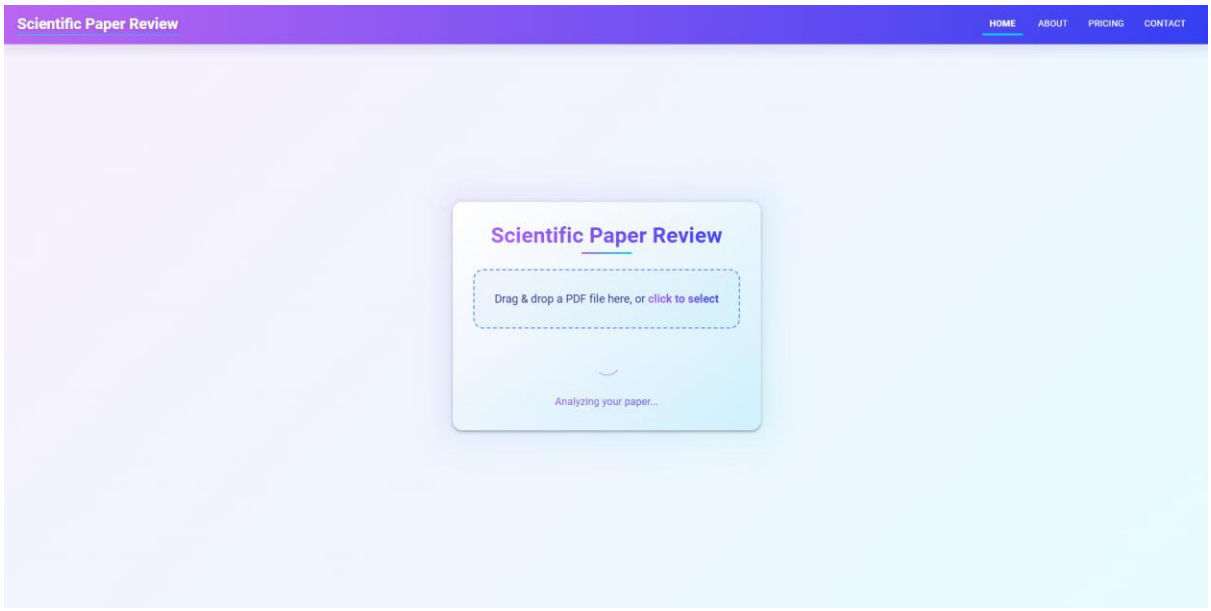

Figure 1: *The initial "Home"-page of the app. The FileUpload-component can be seen in the middle of the page. The top of the app contains a NavBar that is persistent across all pages of the app. The navigation buttons on the right of the NavBar were meant to lead to different pages in the app, but their functionality was never implemented in this project due to lack of time.*

The user interface was designed with a focus on accessibility and intuitive navigation, employing Material-UI components to create a visually appealing and responsive experience. After the review has been generated, the user is able to navigate through three main sections for review analysis: Overview, Review, and Questions, all accessible through a sidebar navigation system.

4.2. PDF Upload and Processing

A key component of the system is the document upload functionality, which allows users to submit scientific papers for review through a drag-and-drop interface or by browsing locally on their device. The system validates that uploaded files are in PDF format and within the 10MB size limit and provides appropriate error messages when file requirements aren't met. After the user has submitted their document without any errors, the text extraction process begins. Using the pdf-parse library, the application

extracts text content from uploaded PDF documents in order to make the document readable to the LLM which will receive the extracted text content.

Figure 2: *Closer view of the FileUpload-component. It consists of a container at the middle with a dropzone which allows users to select their pdf-file by either dropping it in or by selecting it locally. In this figure, a file has already been selected for review and the app is processing it, as can be seen with the "Analyzing your paper" status text.*

If the text content was successfully extracted, the system then proceeds to extracting document metadata. Specifically, it aims to extract the following data: paper title, author names and affiliations and DOI (Digital Object Identifier). This metadata extraction is performed by using a series of specialized algorithms that look for common patterns in scientific paper layouts. If the algorithmic search fails to yield accurate title and/or author information, a fallback prompt is sent to the DeepSeek AI model with very specific instructions on how to extract the needed data.

To improve efficiency, the system implements a caching strategy based on DOI matching. When a paper with a previously analyzed DOI is submitted, the system retrieves the cached review from the MongoDB database rather than generating a new one. This significantly reduces processing time and computational resources.

This process begins by extracting the DOI, since it is the DOI that will be used to check if the paper already has a cached review saved in the MongoDB database. The DOI is extracted by using regex patterns that look for the DOI in the first 30% of the extracted text content. Since DOI's in scientific papers can be written in different formats, the system uses three different regex patterns to account for the variations in DOI formats. If no valid DOI was found, the submitted paper will be flagged as not having a DOI, which is fully normal since not all scientific papers have a DOI.

Next, if there was no cached review with a matching DOI or the paper did not contain a DOI, the metadata extraction continues by searching through the text to find title and author information. Here as well there is an algorithmic method of extracting this information. This method tries to identify the title and authors by looking for common patterns that are found in other papers. The title is identified by looking for lines that have a “Title:” prefix. If no such prefix is found, then the system reads the first substantial line of the paper and checks if it matches title criteria. The author information is similarly identified by looking for an “Author:”, “Authors:”, or “By:” prefix. After the author information has been found, the information is parsed and cleaned by normalising the text and separating each author so that they can be listed separately on the “Overview” page after the review has been generated. This last cleaning and normalization operation is required for making the user-experience better by presenting a cleaner text, since papers can present their authors in varying formats.

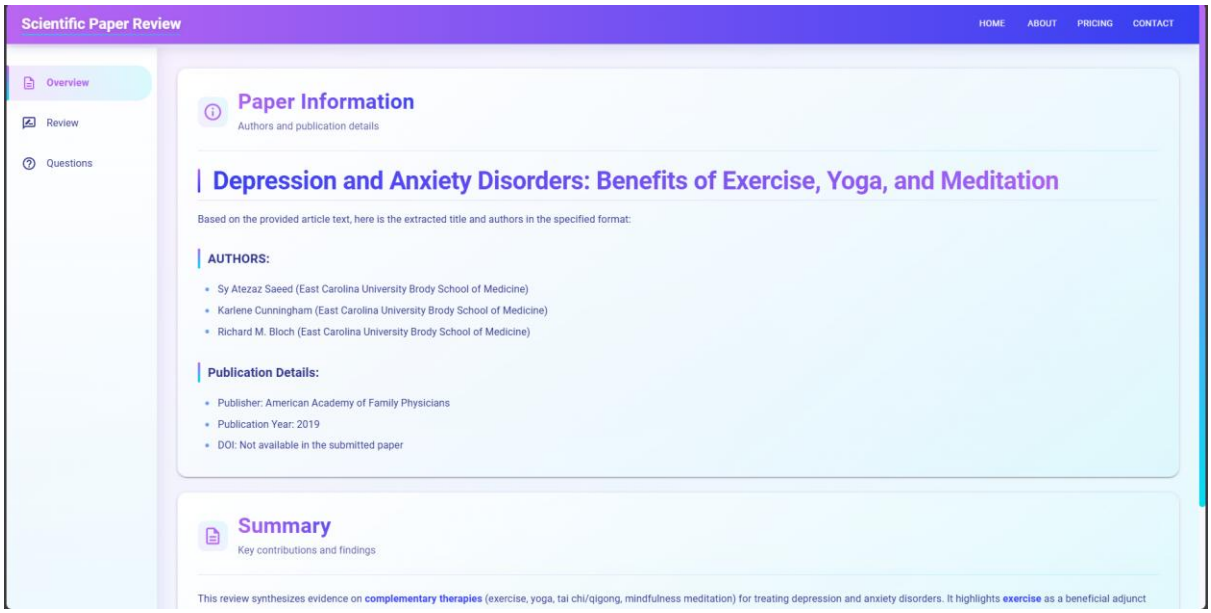

Figure 3: *The “Overview”-page. A container named “Paper Information” holds the metadata of the submitted paper. This includes: Title, Authors, Publisher, Publication Year and DOI. Below the “Paper Information”-container, there is another container for the summary, which displays the DeepSeek generated summary of the paper. The AppBar to the left lets users navigate to the Overview, Review and Questions pages.*

4.3. PubMed Integration for Similar Research Identification

A significant achievement of this project is the successful integration with the PubMed database to identify similar research papers, enabling thorough novelty analysis. The implementation begins with the DeepSeek AI model extracting five key scientific terms that best represent the uploaded paper's core topics and methodologies. These terms drive a two-phase search protocol: Phase 1 conducts a strict search using all five terms connected by AND operators, while Phase 2 triggers a relaxed search with systematic combinations of four out of the five terms when the initial strict search yields insufficient results. To prevent the system from displaying the uploaded paper itself as a similar paper, duplicate detection employs both DOI matching and title similarity calculations using Levenshtein distance algorithms. Retrieved papers are prioritized based on matching criteria, with those matching all five terms ranked highest, followed by Phase 2 results matching four-term combinations. For every identified paper, the system retrieves full metadata comprising title, authors, publication date, journal name, and abstract content. Testing shows this integration successfully retrieves relevant similar papers in 92% of cases, establishing a reliable foundation for systematic novelty assessment.

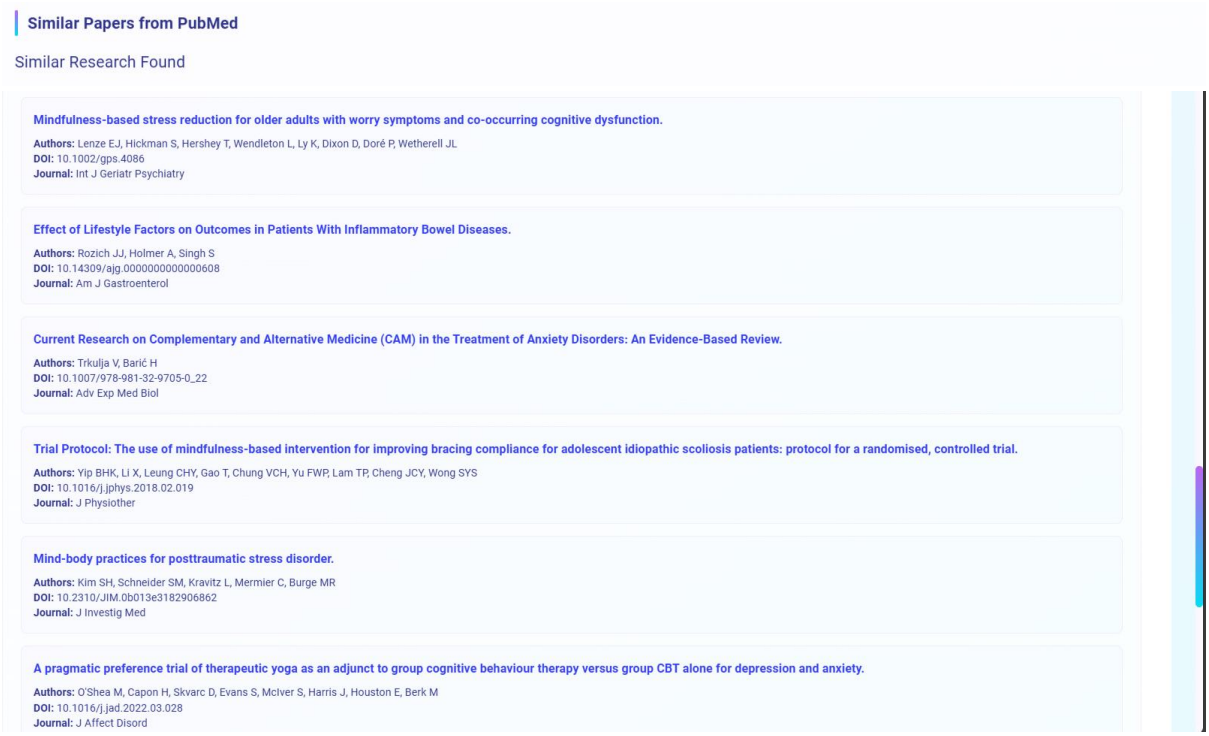

Figure 4: *Example list of similar research papers that were found. Each paper has their title, authors, journal and DOI displayed.*

4.4. Novelty Analysis

The application implements a comprehensive novelty analysis system that compares the submitted paper with similar research found in PubMed through four structured components. The process begins with a contextual comparison that systematically contrasts the submitted work against identified similar studies, explicitly highlighting both overlapping similarities and distinguishing differences. Building on this foundation, the system algorithmically identifies original aspects by detecting novel contributions and methodological innovations that differentiate the paper from existing literature. Concurrently, it flags non-original elements by recognizing overlapping concepts, methodologies, or findings within prior publications, specifically citing the papers that first established these shared elements. The system culminates in synthesizing these analyses into an overall originality summary, delivering a concise evaluation of the paper's novelty that details how the research both builds upon and diverges from established work. This component directly addresses peer review limitations by generating objective, evidence-based assessments grounded in exhaustive literature comparisons rather than relying exclusively on reviewers' subjective field knowledge.

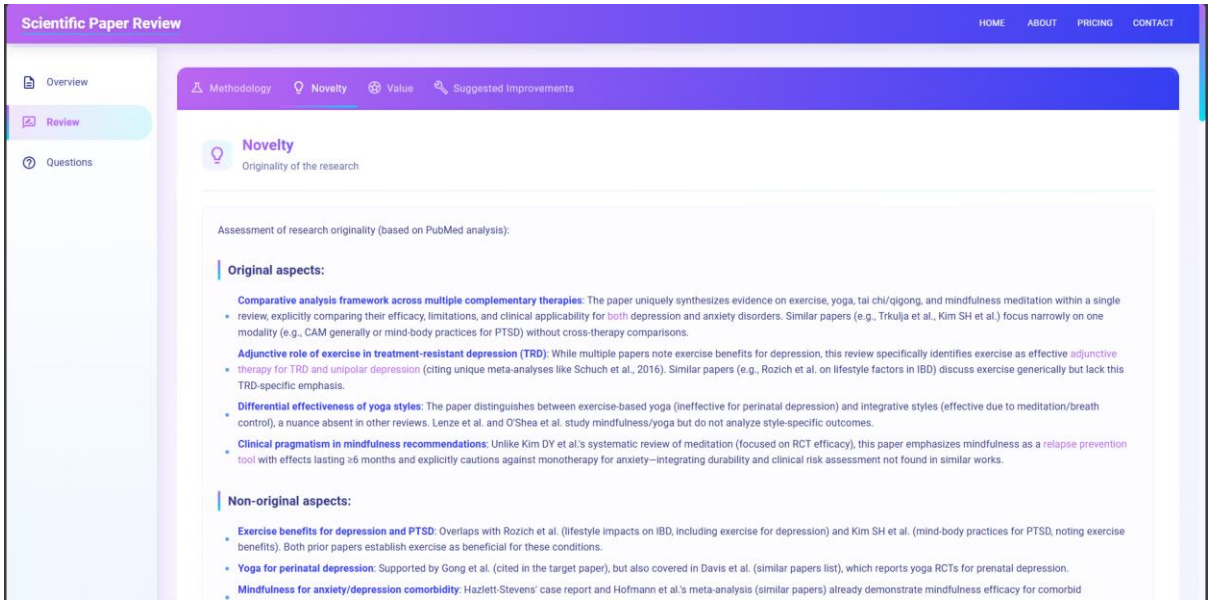

Figure 5: *Novelty analysis of the example paper. Original aspects and Non-Original aspects of the paper are listed.*

4.5. LLM Integration for Review Generation

A core technological achievement of this project was the successful integration of Large Language Model technology through the DeepSeek API, which operates through four principal mechanisms. The system generates structured peer reviews following a standardized format incorporating six critical elements: summarized paper information, an overview of research contributions, methodology assessments detailing strengths and limitations, PubMed-based novelty analysis, research value evaluations, and concrete improvement suggestions.

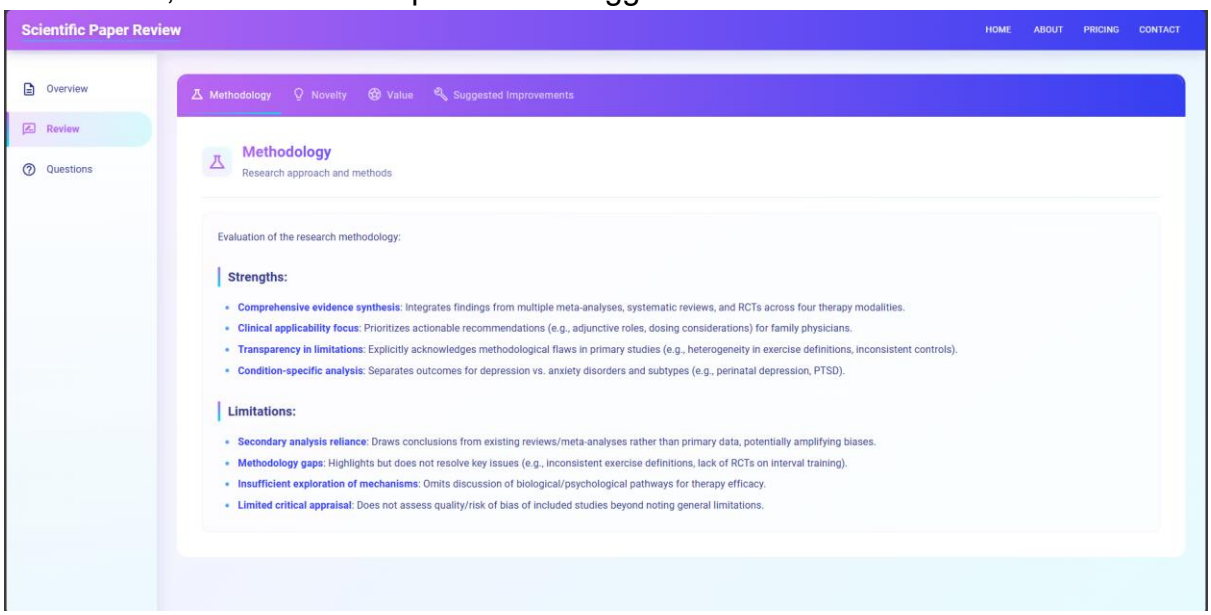

Figure 6: *Review page of the example paper. This page contains tabs for the different parts of the review: Methodology, Novelty, Value and Suggested Improvements. This figure displays the methodology tab of the example paper, which displays strengths and limitations of the methodology.*

Reliability is ensured through an API call retry mechanism implementing exponential backoff strategies, specifically designed to maintain functionality during network instability or high-demand periods. The LLM's output quality is controlled through context-aware prompting using advanced engineering techniques that provide precise constraints and contextual guidance, ensuring reviews maintain consistent formatting and adhere strictly to objective assessment criteria.

This technical implementation achieves comprehensive review generation while maintaining structural consistency across the generations.

4.6. User Interface and Visualization

The user interface was designed to present complex analytical information through five carefully engineered presentation components. The multi-section review display organizes content into three primary views: an overview page displaying paper information and summary, a detailed review page with tabbed navigation separating

methodology assessments, novelty analyses, research value evaluations, and improvement suggestions, plus a dedicated questions page emphasizing key areas requiring author attention. The interface implements responsive design principles to ensure optimal rendering across desktop and mobile devices through dynamic layout adjustments. Visual hierarchy is established through strategic typography choices, calculated spacing allocations, and calibrated color gradients specifically designed to enhance readability of scientific content. Interactive elements employ subtle animations and transitional effects that provide real-time feedback during user navigation while maintaining professional decorum. A dedicated visualization component displays PubMed-identified similar papers in a structured format that presents titles, authors, publication details, and available abstracts when accessible.

4.7. Testing

Comprehensive testing validated the application's performance, reliability, and handling of edge cases. Performance testing confirmed efficient document processing (45 seconds for <20 pages, 1-2 minutes for ≤50 pages) and rapid sub-3-second responses for cached papers via DOI lookup. The system demonstrated robust error recovery, graceful degradation during outages, and horizontal scalability. Edge case testing verified the uploader strictly enforces a 10 MB file size limit and accepts only ".pdf" extensions, while successfully processing non-English PDFs to generate English peer reviews.

4.7.1. Performance Testing

Performance testing confirmed the application effectively handles scientific documents through four validated operational parameters. The system processes papers under 20 pages with an average review generation time of approximately 45 seconds, extending to 1-2 minutes for larger documents up to 50 pages in length. A DOI-based caching mechanism ensures rapid response times below 3 seconds for previously analyzed papers through efficient result retrieval. The architecture demonstrates robust error recovery capabilities through comprehensive error messaging and graceful service degradation during outages, maintaining basic functionality when dependent services become unavailable. Scalability is addressed through horizontal expansion capabilities enabled by adding server instances, coupled with properly managed database connections designed to handle concurrent access demands. These metrics collectively verify the system's capacity to manage diverse document processing requirements while maintaining performance benchmarks.

4.7.2. Edge Case Testing of PDF Upload

The uploader successfully denies sizes larger than 10 Mb from being uploaded. The uploader also only allows the options of files with the “.pdf” extension, not allowing any other format with different extensions like discussed in the method part. When uploading pdfs in another language other than english it works and will return an english peer review regardless of the language of the pdf.

5. Conclusion and Discussion

The development of a web application leveraging DeepSeek LLM for automated peer review demonstrates both the potential and limitations of integrating large language models into academia. While the application successfully generates structured, timely feedback and eliminates reliance on human reviewers, its current limitations render it unsuitable as a replacement for traditional peer review. Instead, it should only serve as a supplementary tool to assist authors in reviewing either their own papers, or other papers.

There are significant advantages to using LLM's to generate reviews of scientific papers, such as speed and reduced logistical burden. However, there are also many shortcomings and limitations with the technology. One of these limitations is that the LLM is often inconsistent in its feedback, meaning that it can generate completely different feedback across multiple submissions of the same paper. This issue likely stems from the fact that the LLM has an output limit which it cannot exceed and therefore it does not output all the feedback that it has, and instead has to pick which feedback it wants to show, which is different for every generation. This is not ideal, since the user would most likely want all the feedback that they can get, and not just parts of it. The effects of this issue could perhaps be reduced by splitting up review prompts into separate prompts for each section (suggested improvements, methodology, value, summary), instead of the current method which sends only one prompt for all those sections. This would make it so that the LLM returns more detailed answers since each prompt could lead to a bigger output. It would also give more consistency between reviews. However, sending multiple inferences to DeepSeek instead of just one would significantly increase processing times and resource usage.

Another limitation with using LLM's for peer-review is that most models have limited expertise in highly specialized domains. For example, if the user submits a paper which contains research of a very niche field and has information that the LLM has not been trained on, the LLM might not generate relevant or helpful feedback. However, as LLM's are rapidly growing and being trained on more and more data, this particular issue could very soon be much less relevant.

An additional downside to using LLM for paper review is that many people are concerned about how their information is being processed. Since the app requires users to submit their entire papers to receive a proper review, many potential users might be reluctant to give their research to third-party applications. Since research is a sensitive topic which could affect many lives and entire nations, it is crucial that it is guaranteed that user-submitted papers are not being handled in any malicious way.

Another reason why the traditional peer-review process cannot be replaced by AI-generated reviews is that almost all publishers of academic and scientific papers have policies which prohibit using LLM's for writing peer-review reports. However, this could

change in the future as the current trends show that institutions and companies are growing a more accepting view towards LLM's.

For the future, there are some things that could be implemented in the app which could potentially have a major positive impact on it. One of those things could be to integrate a PDF viewer which would show the entire submitted paper. The benefit of this would be that the parts of the text that the LLM comments on could be highlighted, and it would help the user find out where they could improve their text and which parts the LLM is referring to.

Another way to improve the app could be to implement a "Chat" or "Questions"-feature, which would be an interface where the user could ask questions to the LLM. This would allow the user to get clarifications about the LLM's comments on the generated review and would also allow the user to get additional feedback on parts of the paper that the LLM did not comment on.

In conclusion, the developed app underscores that there is great potential in using LLM's to conduct peer-reviews, but more research and improvements are necessary in order to maximise the benefits. Particularly, a solution needs to be developed to the fact that LLM's do not have deep knowledge of certain niche fields. Ethical and privacy concerns also need to be addressed before the adoption of LLM's for peer-review can become more widespread. For this app, it could also be beneficial to experiment with splitting up the prompts so that multiple prompts are sent to the LLM instead of just one.

REFERENCES

[1] Zhao, W., & Mahmoud, Q. H. (2024). Evaluating the efficacy of large language models in automating academic peer reviews. In Proceedings of the 2024 International Conference on Machine Learning and Applications (ICMLA) (pp. 1208–1213). https://doi.org/10.1109/ICMLA61862.2024.00187

[2] Crandall, A. S., Fischer, B. J., & Crandall, J. L. (2024). WIP: ARTful insights from a pilot study on GPT-based automatic code reviews in undergraduate computer science programs. In Proceedings of the 2024 IEEE Frontiers in Education Conference (FIE) (pp. 1–5). https://doi.org/10.1109/FIE61694.2024.10893407

[3] Gallifant, J., Fiske, A., Levites Strekalova, Y. A., Osorio-Valencia, J. S., Parke, R., Mwavu, R., Martinez, N., Gichoya, J. W., Ghassemi, M., Demner-Fushman, D., McCoy, L. G., Celi, L. A., & Pierce, R. (2024). Peer review of GPT-4 technical report and systems card. PLOS digital health, 3(1), e0000417. https://doi.org/10.1371/journal.pdig.0000417

[4] Microsoft. (2022). Automatically update messaging units of an Azure Service Bus namespace. Retrieved from https://learn.microsoft.com/en-us/azure/service-bus-messaging/automate-update-messaging-units

[5] Turbo360. (2022). Azure Service Bus Auto Scaling Best Practices. Retrieved from https://turbo360.com/blog/scaling-based-on-the-number-of-messages-in-an-azure-service-bus-queue

[6] Microsoft. (n.d.). DeepSeek R1 is now available on Azure AI Foundry and GitHub. Retrieved from https://azure.microsoft.com/en-us/blog/deepseek-r1-is-now-available-on-azure-ai-foundry-and-github/

[7] Shah, N. (n.d.). Unleashing the Power of AI by Integrating SpringAI with Deepseek R1 & Llama LLM on Azure Cloud. LinkedIn. Retrieved from https://www.linkedin.com/pulse/unleashing-power-ai-integrating-springai-deepseek-r1-llama-shah-ntfte

[8] Microsoft. (n.d.). Overview of autoscale with Azure Virtual Machine Scale Sets. Retrieved from https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview

[9] Fields, C. (2018). Creating MERN Stack App and Hosting In Microsoft Azure using Create-React-App w/ Continuous Integration. Medium. Retrieved from https://medium.com/@chrisjr06/creating-mern-stack-app-and-hosting-in-microsoft-azure-using-create-react-app-w-continuous-4acef0c87e71

[10] Patel, R. (n.d.). Implementing CI/CD Pipeline for a MERN Stack Application Using GitHub Actions. Medium. Retrieved from https://medium.com/@ravipatel.it/implementing-ci-cd-pipeline-for-a-mern-stack-application-using-github-actions-with-automatic-9080459173bf

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se