MathTextor: Humanizing Mathematical Typesetting for Undergraduate Students

Minseok Kim
Computer Science, NYUAD
mk7545@nyu.edu

Advised by: Moumena Chaqfeh (moumena@nyu.edu), Steven Euijong Whang (swhang@kaist.ac.kr)

ABSTRACT

Mathematical writing presents a significant challenge for students due to its complex notation and strict formalism. Two major obstacles are the steep learning curve associated with mastering mathematical typesetting languages (such as LaTeX) and the considerable time investment required to produce professional, math-heavy documents. While Large Language Models (LLMs) have been increasingly adopted to support academic writing, existing tools often lack the contextual support needed to efficiently produce well-structured mathematical expressions.

This research presents **MathTextor**, an AI-powered writing interface designed to humanize digital mathematical writing. MathTextor integrates a contextual AI component to automatically suggest relevant mathematical symbols based on the problem context, reducing the need for rote memorization of syntax. We evaluated the system through a user study with 24 undergraduate participants comparing three methods: handwriting, LaTeX, and MathTextor. Results demonstrate that while handwriting remains the fastest method, MathTextor significantly outperforms LaTeX in terms of speed for novice users. Specifically, for slower typists, MathTextor was on average 38.63% faster than LaTeX. Furthermore, participants reported lower mental and temporal demand when using MathTextor compared to standard LaTeX editors. This project demonstrates that context-aware AI tools can effectively bridge the gap between handwritten intuition and digital formalism.

KEYWORDS

LaTeX, Large Language Model, Generative AI

Reference Format:
Minseok Kim. 2025. MathTextor: Humanizing Mathematical Typesetting for Undergraduate Students. In *NYUAD Capstone Seminar Reports, Spring 2025, Abu Dhabi, UAE.* 9 pages.

1 INTRODUCTION

Undergraduate students encounter two intertwined challenges when writing mathematics: (1) the cognitive overhead of translating mathematical ideas into formal notation, and (2) the steep learning curve of mastering LaTeX syntax for those not already fluent. While LaTeX is the gold-standard for producing professional-quality documents in mathematics, physics, and engineering, its extensive symbol set and markup conventions impose a significant barrier: students must memorize commands like `\forall`, `\sum_{i=1}^n`, or `\mathbb{R}` before they can focus on solving problems. This can lead to frustration, errors in notation, and lost time that could otherwise be devoted to conceptual understanding and problem-solving.

The importance of lowering this barrier is twofold. First, by reducing the syntax burden, we empower students to focus on mathematical reasoning rather than typesetting minutiae. This has the potential to improve learning outcomes in any course that relies heavily on written mathematics—from discrete mathematics and linear algebra to real analysis and differential equations. Second, instructors and graders also benefit: when student submissions are generated with consistent, unambiguous notation, grading becomes faster and more reliable, and the risk of misunderstanding poorly formatted work is minimized.

Research Questions: This study addresses the following questions:

(1) To what extent can AI components aid mathematical writing?
(2) What benefits can these components bring to existing mathematical interfaces?

Recent advances in generative AI—especially Large Language Models (LLMs)—have demonstrated remarkable capabilities in translating between natural language and formal representations, including code, SQL queries, and even LaTeX math. However, existing integrations of LLMs into writing tools often focus on prose generation or generating isolated equations lacking contextual awareness of the learner's current problem set or the structure of an entire assignment. As a result, they do not fully eliminate the need for learners to know LaTeX commands or to intervene into the raw model outputs for formatting.

Current solutions collectively reveal a critical gap: no existing system provides an intelligent, context-aware generation of mathematical notation. They fall into three incomplete categories:

(1) Passive converters (OCR, image-to-LaTeX)
(2) Manual editors (syntax-specific)
(3) Lack of contextual understanding (static recommendation of symbols)

This project aims to bridge this gap by combining:

- Context-aware AI that suggests notation based on the solution for each problem
- An interactive toolbar interface inspired by educational platforms

Unlike previous approaches that treated interpretation and generation as separate challenges, our integrated approach addresses both the understanding and generation aspects of mathematical writing.

In this project, we propose **MathTextor**, an AI-powered interface that seamlessly converts instructor-provided mathematical questions into fully formatted LaTeX expressions for students. When questions are uploaded to our UI, MathTextor will—behind the scenes—(1) parse the problem statement, (2) invoke an integrated LLM to generate the precise LaTeX symbol lists and structural markup required for each solution, and (3) present ready-to-insert LaTeX snippets that students can use directly in their solutions. By bridging the gap between problem comprehension and LaTeX syntax, MathTextor removes rote memorization of commands from the workflow, reduces syntax errors, and accelerates the pace of mathematical composition.

The remainder of this paper is organized as follows: Section 2 reviews related work in AI-assisted mathematical typesetting and educational tools; Section 3 details our methodology, including system architecture, LLM selection, and UI design; Section 4 describes our evaluation plan and user study; Section 5 presents the quantitative and qualitative results; and Section 6 concludes with contributions and broader impacts.

2 RELATED WORK

Computational approaches to mathematical writing have long faced significant challenges in both interpretation and generation of formal notation. Early efforts to address these challenges focused on optical character recognition (OCR) systems like InftyReader, which could extract handwritten or printed mathematical symbols into digital formats but lacked generative or contextual editing capabilities [3]. The emergence of LaTeX provided a powerful typesetting solution, but introduced new barriers through its complex markup syntax that requires extensive memorization of commands. Subsequent tools like MathType and LyX attempted to simplify equation editing through graphical interfaces, yet still required manual symbol selection and offered no intelligent assistance [6, 8].

Recent advancements in AI and large language models have created new opportunities for mathematical writing assistance. While tools like Mathpix improved OCR-to-LaTeX conversion [2] and MathDeck and Wolfram Alpha enhanced structural editing with toolbar [1, 4], they remain limited to either static image input or manual query-based workflows. Modern LLM integrations like TeXGPT demonstrate the potential of AI for LaTeX syntax, but fail to provide domain-specific support for mathematical notation [5].

By building on design principles from Weisz et al. [7] while incorporating insights from existing math text editors, MathTextor offers the first unified solution for computationally-assisted mathematical writing that is both intelligent and contextual.

3 SYSTEM ARCHITECTURE

We define two potential users for MathTextor:

- **The Author**: The user who creates math problems, exercises, or questions using the author interface. Typical users include instructors and content creators for learning platforms.
- **The Solver**: The user who constructs and submits mathematical solutions in response to problems created by the author. In the AI-assisted toolbar interface, the solver receives context-aware symbol suggestions and structural templates—minimizing manual typesetting. Typical users primarily include students, self-learners, or peer learners reviewing or answering problems.

3.1 Data Flow Diagram

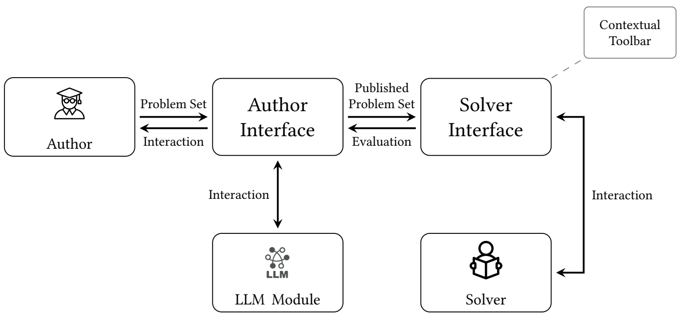


Figure 1: Enhanced data flow diagram showing complete interaction cycle with LLM integration

3.2 Overall Description

The MathTextor system consists of three main components arranged in a modular web-based architecture (Figure 2):

(1) **Web Frontend:** A client-side interface built with **HTML and JavaScript** that serves as the entry point for both authors and solvers. It manages the DOM to display problem sets and renders the AI-generated LaTeX snippets directly in the browser.
(2) **LLM Backend:** A server-side module acting as an API gateway. It receives problem context from the frontend, constructs prompts using pre-defined templates, and queries the Gemini 2.5 Pro model to extract mathematical symbols.
(3) **Evaluation Engine:** A backend component designed to ensure the robustness and quality of the generated outputs to ensure reliability and precision.

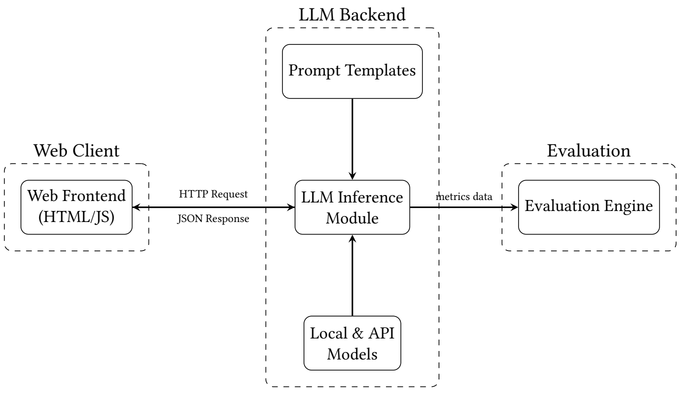
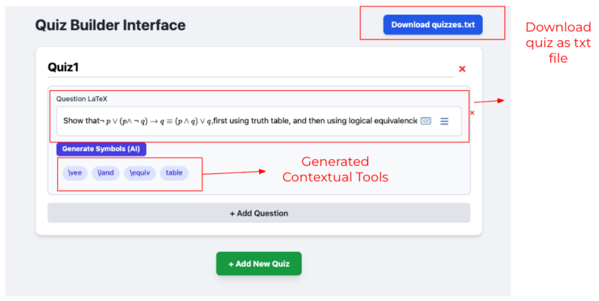


Figure 2: High-Level Architecture of MathTextor showing the interaction between the Web Client and the LLM Backend.

3.3 The Author Interface

The Author Interface provides the authors with a dedicated workspace to:

- Input mathematical problem sets using the input field
- Review and validate problem formulations
- Organize and publish finalized problem sets



Show that $\neg p \vee (p \wedge \neg q) \rightarrow q \equiv (p \wedge q) \vee q$,first using truth table, and then using logical equivalencie

Figure 3: The Author Interface (Quiz Builder) where instructors input problem statements. The system automatically generates the AI symbol context shown in the preview.

This interface serves as the entry point for creating the mathematical content that the solvers will engage with through the Solver Interface. All published problem sets maintain the original structure and notation provided by the author.

3.4 The Solver Interface

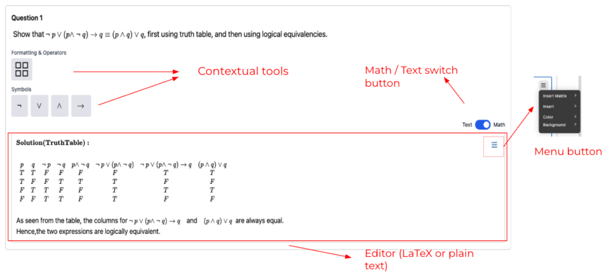
The Solver Interface provides the solvers with key UI elements including:

- **Solution Input Panel**
  - Provides tools for constructing solutions and take input solutions for each problem
- **Snippet Preview**
  - A live preview snippet rendering the generated LaTeX expression.
- **AI-Assisted Toolbar**
  - Context-aware mathematical notation suggestions
- **Submission Controls**
  - Buttons to copy the constructed solutions in LaTeX code to clipboard or save it to a `.pdf` file.

This interface maintains focus on mathematical problem-solving while reducing syntactic barriers through intelligent assistance.

4 METHODOLOGY

In this section, we describe the design and implementation of MathTextor, our AI-powered mathematical writing assistant. We first present the overall system architecture, then detail


Question 1

Show that $\neg p \vee (p \wedge \neg q) \to q \equiv (p \wedge q) \vee q$, first using truth table, and then using logical equivalencies.

Solution(TruthTable) :

p	q	$\neg p$	$\neg q$	$p \wedge \neg q$	$\neg p \vee (p \wedge \neg q)$	$\neg p \vee (p \wedge \neg q) \to q$	$(p \wedge q) \vee q$
T	T	F	F	F	F	T	T
T	F	F	T	T	T	F	F
F	T	T	F	F	T	T	T
F	F	T	T	F	T	F	F

As seen from the table, the columns for $\neg p \vee (p \wedge \neg q) \to q$ and $(p \wedge q) \vee q$ are always equal.
Hence,the two expressions are logically equivalent.

Figure 4: The Solver Interface showing the AI-generated symbol suggestions (e.g., ∨, ∧, →) tailored to the specific logic question.

the **web technologies (HTML/JavaScript)** used for the UI, the integration of Large Language Models (LLMs), and finally outline our experimental procedure for model evaluation and selection.

4.1 Design Considerations

For the MathTextor interface, we adopted four core design principles from Weisz et al. [7] relevant to our use case:

1) **Content Creation**: Author interface provides AI generation for formatting and mathematical symbols as well as an editing area that converts the input into LaTeX and presents rendered notations.

2) **Appropriate Trust**: All AI-generated content will appear with a gray background and require explicit Author approval before being published and shown to learners, helping prevent overreliance.

3) **Contextual Variability**: The toolbar's suggested symbols will dynamically adapt to problem types (e.g., to the problem "How can this English sentence be translated into a logical expression? "You can access the Internet from campus only if you are a computer science major or you are not a freshman." the toolbar will show $\to, \vee, \neg$), with visual grouping to maintain consistency.

The design will follow the common toolbar organization in mathematical text editors, with:

- Context-aware symbol toolbar tailored to each problem entry.
- One-click insertion with live LaTeX preview
- Persistent editing capability for all generated content

4.2 LLM Integration and Performance

We initially explored both local models and API-based models. Testing revealed that lightweight local models struggled with complex mathematical reasoning and context extraction. Consequently, the final system utilizes the **Gemini 2.5 Pro API**. The evaluation section entails the details of the model performance evaluation.

The AI component was designed to analyze LaTeX text from the problem statement and extract unique mathematical symbols required to write the solution. The model was instructed via a strict system prompt to return a valid JSON array of strings containing only symbols from a predefined valid list. To ensure output stability, the prompt enforced specific constraints: (1) strict adherence to the provided whitelist of symbols, (2) exclusion of any conversational text or markdown formatting, and (3) fallback to an empty array if no symbols were detected. Additionally, we employed few-shot prompting, providing the model with input-output examples (e.g., mapping logical expressions like $\neg(A \wedge B)$ to their constituent LaTeX commands) to standardize the JSON structure.

4.3 Component Evaluation

To validate the AI component before the user study, we utilized a dataset of 50 manually annotated Discrete Mathematics problem-symbol dataset. The performance metrics for the Gemini 2.5 Pro integration were:

- **Precision:** 0.7160
- **Recall:** 0.7420
- **F1 Score:** 0.7288
- **Latency:** $<$ 10 seconds

These metrics confirmed that the model could reliably generate the necessary symbol context for the toolbar.

4.4 Reproducibility and Implementation Details

All code is version-controlled in a Git repository. The **web application** source code and evaluation scripts include configuration files specifying model paths, API keys, and evaluation parameters.

5 USER STUDY

We conducted an IRB-approved user study to evaluate the effectiveness of MathTextor in comparison to traditional handwriting and standard LaTeX editing (Overleaf).

5.1 Participants and Setup

The study included 24 undergraduate participants aged 18-24. Participants were selected based on the criteria of having completed a Discrete Mathematics course and having proficiency in LaTeX. This selection criteria was chosen to ensure a rigorous comparison, effectively placing MathTextor at a disadvantage by testing it against users already comfortable with the competing digital tool.

5.2 Procedure

The study followed a within-subjects design. To ensure a rigorous evaluation, the study design purposefully placed the proposed tool at a specific disadvantage regarding familiarity and ordering:

- **Low Familiarity:** Unlike their established proficiency in LaTeX, participants received only a 5-minute training session on the new interface.
- **First Ordering:** The web-based interface was always used first. This meant participants spent more time thinking and conceptualizing the solution during this task, whereas subsequent tasks (Handwriting and LaTeX) involved transcribing a solution they had already formulated.

To mitigate potential novelty bias, the prototype was not referred to by its specific name (MathTextor) during the session; it was introduced simply as a "web-based interface."

The experimental procedure consisted of the following steps:

1. **Typing Test:** Participants completed a brief typing assessment to establish a baseline for their mechanical typing speed.
2. **Training:** Participants received 5 minutes of training on the MathTextor solver interface to learn the toolbar and input mechanisms.
3. **Problem Materials:** Participants were given two Discrete Math problems (Truth Table and Set Theory) and an assistance sheet containing standard logical equivalences.
4. **Thinking Time:** A dedicated period was allotted for participants to think through the solutions and write preliminary notes before interacting with any tool.
5. **Web-based Interface Task:** Participants wrote the solutions to both problems using MathTextor.
6. **Handwriting Task:** Participants wrote the same solutions by hand on paper.
7. **LaTeX Task:** Participants typed the same solutions in Overleaf using a provided LaTeX template (with the problem statements pre-typed).
8. **Surveys:** Participants completed a workload survey after each method, followed by a final comprehensive survey at the end of the study.

Figure 5 shows the 2 simple logic and set theory questions used for the user study.

6 RESULTS AND DISCUSSION

6.1 Quantitative Results

The timing analysis revealed a clear hierarchy in speed: $Time(Handwriting) < Time(MathTextor) < Time(LaTeX)$. While handwriting remains the fastest medium for mathematical thought, MathTextor demonstrated significant efficiency gains over LaTeX for specific user groups.

Question 1 (Logic):
Show that $\neg p \vee (p \wedge \neg q) \rightarrow q \equiv (p \wedge q) \vee q$, first using truth table, and then using logical equivalences.

Question 2 (Set Theory):
Prove or disprove: For any sets A and B, $\overline{(A - B)} = \overline{A} \cup \overline{B}$

Figure 5: The specific Discrete Mathematics problems assigned to participants during the user study.

Impact of Typing Speed: For faster typists (> 46 WPM), the difference between MathTextor and LaTeX was less pronounced. However, for slower typists (< 46 WPM), MathTextor was significantly faster, reducing completion time by an average of **38.63%** compared to LaTeX. This suggests that MathTextor successfully lowers the barrier to entry for students who are less mechanically proficient with typing code.

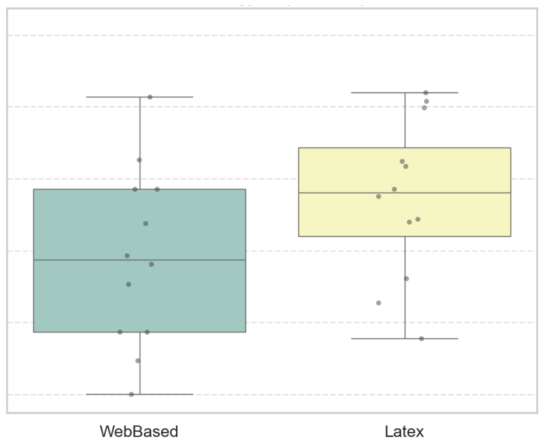


Figure 6: Time distribution comparison for slower typists (< 46 WPM), showing a significant speed advantage for MathTextor (WebBased) over LaTeX.

Device Familiarity: Participants using their own laptops performed 30.6% faster on MathTextor compared to LaTeX, whereas the gap narrowed on lab PCs. This indicates that the web-based interface is sensitive to the user's input device ecosystem (e.g., trackpad gestures, screen resolution). The contextual toolbar typically requires more screen real estate, which may have impacted the visual scan path on standard lab monitors compared to personal devices.

6.1.1 *Survey Data Analysis.*

Participant feedback highlighted the "compromise" nature of MathTextor. Users noted that it combines the structure of a digital editor with the ease of a visual interface.

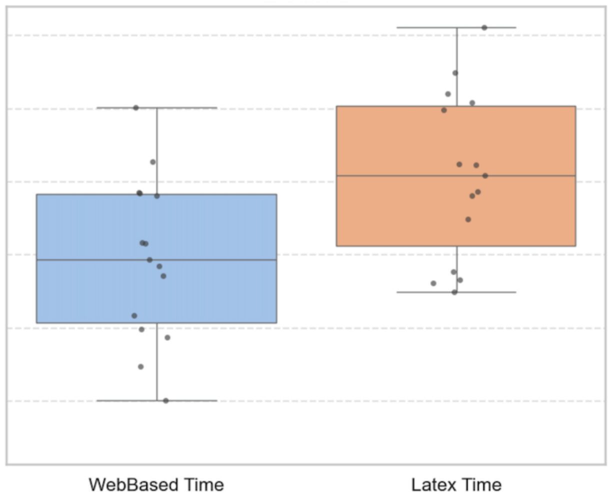
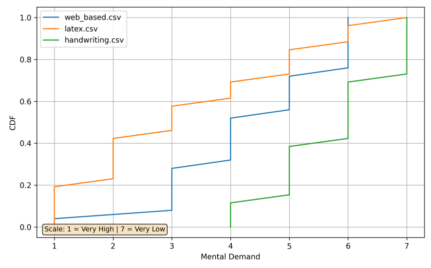
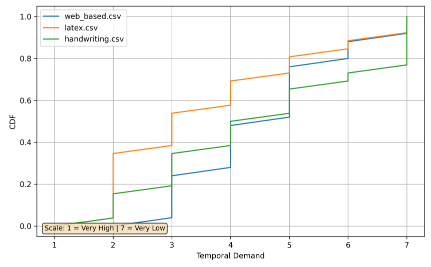


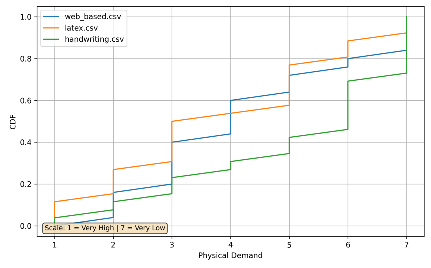
Figure 7: Time comparison for participants using their own laptops. MathTextor demonstrated a 30.63% speed advantage over LaTeX when users were familiar with their hardware.



(a) Mental Demand



(b) Temporal Demand



(c) Physical Demand

Figure 8: CDF of Primary Task Load Metrics. Higher scores (closer to 7) indicate lower demand. MathTextor achieves higher score in having low demand in all three demands.

Scale Interpretation: For all reported metrics below, the scale is normalized such that **1 represents the most negative outcome** (e.g., Very High Demand, Very Difficult, Total Failure) and **7 represents the most positive outcome** (e.g., Very Low Demand, Very Easy, Perfect Performance).

Task Load and Effort: We analyzed specific dimensions of workload to understand the user experience:

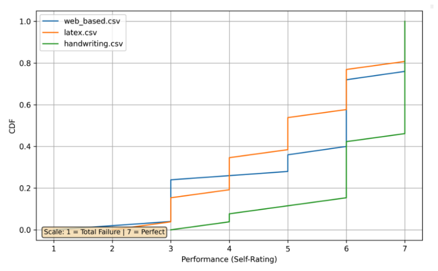
- **Primary Demands (Fig. 8):** LaTeX imposed the highest cognitive and temporal load. MathTextor consistently bridged the gap, shifting the distribution closer to the natural baseline of handwriting.
- **Effort and Frustration (Fig. 9):** While LaTeX caused the highest frustration, MathTextor performed comparably to handwriting in terms of self-rated performance and required effort, indicating a smoother user journey than traditional coding.

Usability and Learning: Beyond workload, we evaluated specific usability factors (Fig. 10):

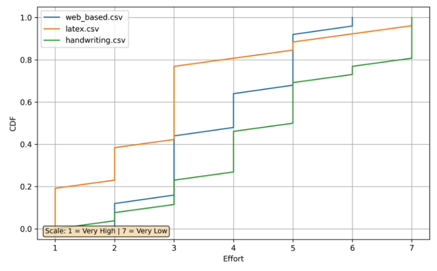
- **Ease of Use & Accuracy:** MathTextor scored higher on ease of use than LaTeX. Notation accuracy confidence was high, suggesting users trusted the AI suggestions.
- **Distraction & Learning:** MathTextor minimized distractions compared to the syntax-heavy LaTeX environment. The learning curve for MathTextor was also rated favorably, approaching the intuitive nature of handwriting.

Overall Summary: The comparative analysis is summarized in the Radar Chart (Fig. 11a), which visualizes how MathTextor envelopes LaTeX across almost all positive dimensions. Finally, the Overall Average Rating (Fig. 11b) confirms that while Handwriting remains the most preferred method (5.60 ± 1.69), MathTextor (4.57 ± 1.62) successfully outperforms LaTeX (3.95 ± 1.91) as a digital alternative.

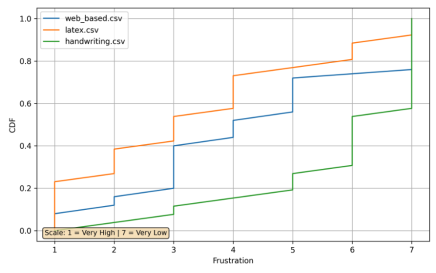
Cognitive Friction of Mode Switching: While users praised the tool as a "perfect compromise", the reported annoyance regarding the "iteration from text to math method" highlights a known HCI phenomenon: *mode error*. The cognitive cost of switching context—moving hands from the keyboard to the mouse to select symbols—creates micro-interruptions in the writing flow. This validates our future work goal of implementing keyboard shortcuts to reduce latency.



(a) Self-Rated Performance



(b) Effort Required



(c) Frustration Level

Figure 9: CDF of Extended Load Metrics. MathTextor reduces required effort and frustration significantly compared to LaTeX; self-reported performance is also higher compared to LaTeX.

6.2 Qualitative Results

Participants provided extensive qualitative feedback through the post-task surveys. This feedback was categorized into three primary themes: the difficulty associated with standard LaTeX, positive reception of the MathTextor interface, and constructive feedback for future improvements.

6.2.1 Barriers of Standard LaTeX. Participants expressed significant frustration with the cognitive load imposed by standard LaTeX editors. The feedback highlights that the syntax often distracts from the mathematical concepts themselves:

- *"...latex took me so much time like 20 minute to write the 2 questions when i wrote it in 2 minute in paper."*
- *"Doing Discrete Mathematics assignments directly in Latex has somehow made me focus more on typing than actually understanding the solutions."*
- *"The coding part is mentally draining."*
- *"Needed a lot of thinking for typing out different latex commands, also to figure out where to put what part of the code."*
- *"latex was incredibly hard to use."*
- *"Working with tables can be messy some times."*
- *"I had to navigate not only the math, but also the platform, learning how to do things with it that are much simpler and faster on paper."*

6.2.2 Positive Feedback for MathTextor. In contrast, MathTextor was frequently described as a bridge between the ease of handwriting and the formalism of digital typing. Participants appreciated the accessibility for beginners and the context-aware tools:

- *"This is like the perfect compromise between handwritten and latex written solutions where we have a online editor which we can use to write out all the symbols etc. without having to write latex."*
- *"Since I prefer digital form of submission, I consider the web interface gives the most convenient medium to deliver submissions; LaTeX would be a little too complicated for assignments."*
- *"There are benefits I can see. For example the presence of simple drag and drop notations makes it looks simpler for a beginner, while also allowing the latex versatility that overleaf offers."*
- *"Overall, I think the Web based interface is really good for question and answer type responses... It's far easier to use sets, and common symbols in the Web based interface."*
- *"I don't like writing, otherwise I would have chosen handwriting. If I have to place answers in a box, I would use the Web-based interface."*
- *"I can see how for beginners the web tool can be helpful."*
- *"I like the quiz like version if its just question and response."*

6.2.3 Constructive Feedback and Future Features. While the reception was positive, participants identified friction points regarding the workflow, specifically the switching between text and math modes. They also requested features that would further emulate the freedom of handwriting:

- **Workflow Friction:** *"The most noticeable inconvenience was the text-math toggle, as the behavior was not fully consistent."* Another user noted, *"Software was limiting in options, and the iteration from text to math method was annoying."*
- **Feature Requests:** Participants requested *"seemlessness between text and math mode (maybe add a keyboard shortcut!)"* and the *"ability to add diagrams or even scribble in their screen,"* specifically mentioning *"adding ven diagrams or geometry stuff."*

- **Optimism:** despite limitations, users noted, *"I think with practice we get used to the interface and makes it easier,"* and concluded that *"a web-based interface that's a bit more optimized and user-friendly could turn out to be better than LaTeX for smaller purposes that don't need features too advanced."*

6.3 Limitations and Future Work

A primary limitation identified during the user study was the friction associated with mode switching. Participants noted that while the symbol generation was helpful, the cognitive overhead of toggling between text and math entry via the mouse interrupted their writing flow. To address these usability constraints, future development will prioritize:

- **Workflow Optimization:** Implementing global keyboard shortcuts to toggle between text and math modes, significantly reducing latency and mouse dependency.
- **Visual Support:** Extending the generative capabilities to support mathematical diagrams (e.g., Venn diagrams, graph plots), addressing user feedback regarding the need for visual aids.
- **Hybrid Input:** Integrating handwriting recognition features to allow users to scribble input directly on the screen, further bridging the gap between analog intuition and digital formalism.

Model Expansion and Evaluation: Currently, MathTextor relies on a single LLM backend. To enhance the robustness and accuracy of the symbol generation, future iterations will expand the scope of evaluation by integrating and comparing multiple AI architectures. Specifically, we aim to incorporate computational knowledge engines, such as **WolframAlpha**, alongside other Large Language Models. Unlike purely probabilistic LLMs, integrating WolframAlpha could provide deterministic validation for mathematical notation, potentially reducing hallucination rates in complex problem sets. A comparative analysis across these different backends would help identify the optimal balance between generative flexibility and symbolic precision.

7 CONCLUSION

This project introduced MathTextor, a context-aware AI interface for mathematical writing. By leveraging LLM component to generate problem-specific symbol sets, we demonstrated that it is possible to reduce the cognitive load associated with LaTeX typesetting.

Our results show that MathTextor serves as an effective bridge between analog and digital workflows. It significantly outperforms standard LaTeX editors in speed for slower typists and reduces mental demand for users. While handwriting remains the fastest method for pure generation, MathTextor offers a viable, more inclusive digital alternative that lowers the barrier to producing professional mathematical documents.

- **Reduce cognitive load** through context-aware symbol suggestions, allowing students to focus on mathematical reasoning rather than typesetting
- **Enhance educational outcomes** through standardized notation that benefits both learners and graders

Key Innovations:

- First integration of LLMs specifically optimized for mathematical notation
- Dual-interface design separating problem creation and solution workflows
- Context-aware toolbar that adapts to each problem

Broader Impacts:

- Potential applications in STEM education beyond mathematics
- Framework for domain-specific AI writing assistants
- Open-source implementation to support educational accessibility
- Potential extension to Overleaf plug-in

Generalizability: Although the user study was conducted within the specific scope of Discrete Mathematics, MathTextor has the potential to be applied effectively across broader mathematical fields. Because the system utilizes a general-purpose LLM API rather than a model trained on a narrow, field-specific dataset, it is capable of handling complex questions in other domains. Furthermore, the system design ensures flexibility: instructors retain the freedom to manually edit generated questions, and solvers can access a comprehensive library of mathematical symbols via the manual menu button. This ensures that the tool remains robust even in scenarios where the AI's contextual predictions may require refinement.

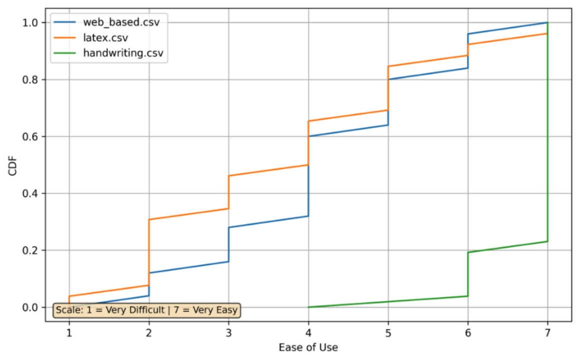
The evaluation conducted with 24 undergraduate students quantitatively measured time savings and accuracy improvements compared to traditional methods. The results confirm that successful implementation of MathTextor fundamentally changes how students interact with mathematical notation, lowering barriers to entry in mathematics while maintaining the rigor of formal mathematical communication.

REFERENCES

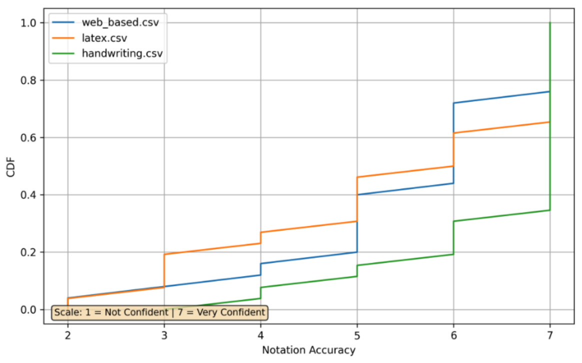
[1] Yancarlos Diaz, Gavin Nishizawa, Behrooz Mansouri, Kenny Davila, and Richard Zanibbi. 2021. The mathdeck formula editor: Interactive formula entry combining latex, structure editing, and search. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–5.

[2] Mathpix. 2025. Mathpix: Image to LaTeX Conversion Tool. https://mathpix.com. Accessed: 2025-05-06.

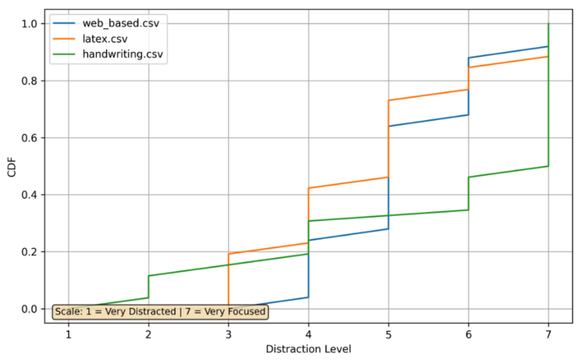
[3] Infty Project. 2020. InftyReader: OCR for Math Documents. https://www.inftyproject.org. Accessed: 2024-11-18.



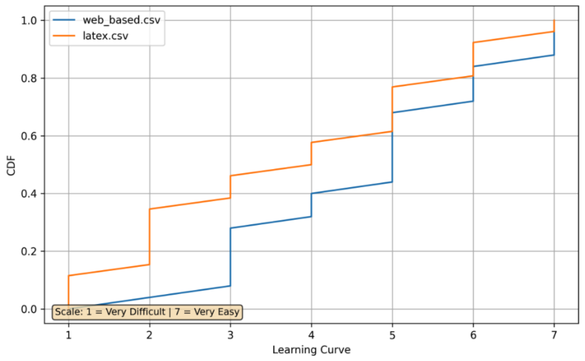
(a) Ease of Use



(b) Notation Accuracy Confidence

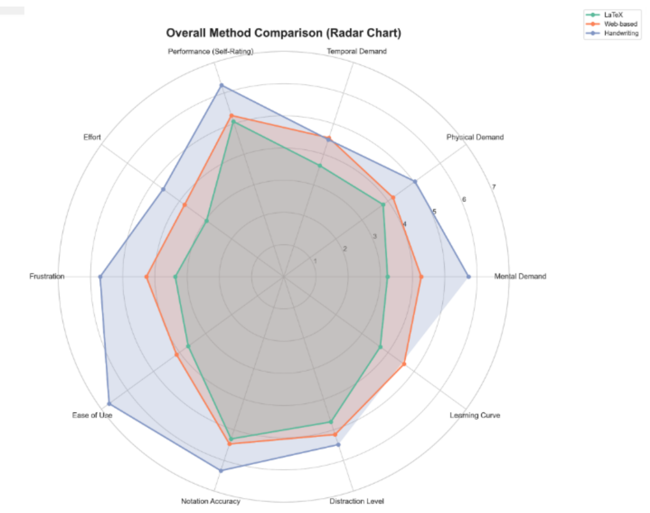


(c) Distraction Level (High=Focused)

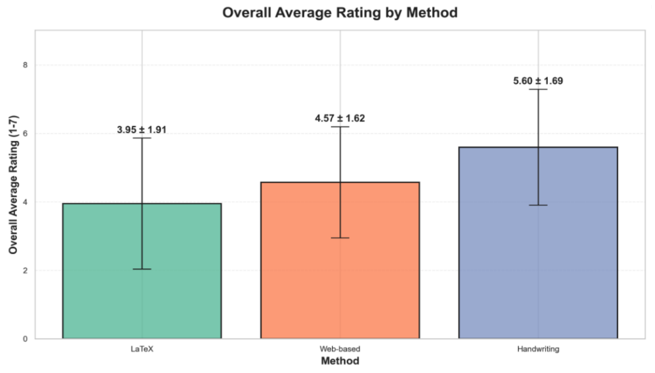


(d) Learning Curve (High=Easy to Learn)

Figure 10: Usability Experience Metrics. MathTextor demonstrates a gentler learning curve and higher ease of use than LaTeX. MathTextor is also reported as having higher notation accuracy and less distracting than LaTeX.



(a) Method Comparison Radar Chart



(b) Overall Average User Rating

Figure 11: Summary of Qualitative Results. The Radar Chart (a) shows MathTextor covering a larger area of positive user experience than LaTeX. The Bar Chart (b) shows the aggregate preference scores.

[4] Wolfram Research. 2023. WolframAlpha: Computational Intelligence. https://www.wolframalpha.com. Accessed: 2024-11-18.

[5] Digital Science. 2024. Enhanced AI tool TeXGPT powers up academic writing. *NISO* (2024). https://www.niso.org/niso-io/2024/04/enhanced-ai-tool-texgpt-powers-up-academic-writing

[6] LyX Team. 2022. LyX: The Document Processor. https://www.lyx.org. Accessed: 2024-11-18.

[7] Justin D Weisz, Jessica He, Michael Muller, Gabriela Hoefer, Rachel Miles, and Werner Geyer. 2024. Design principles for generative AI applications. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–22.

[8] WIRIS. 2022. MathType: Equation Editor with LaTeX Export. https://www.wiris.com/mathtype. Accessed: 2024-11-18.