



INTEGRAL

Interactive Notetaking Tool for Equations, Graphs, Algebra and Logic
Co-creating an AI-Assisted Mathematical Notetaking Platform

Final Project Report

Students-as-Co-Creators Initiative
Learning and Teaching Research Collaborations
Centre for Education and Teaching Innovation (CETI)
University of Westminster

Project Team

Academic Partners

Dr Thomas Oliver
Dr Natalia Yerashenia

Student Partners

Erik Manukyan
Rosh Memari
Matt Petley
Arvin Shafiei
Mantas Zvybas

Project Duration

November 2025 – July 2026

Submitted

July 2026

School of Computer Science and Engineering

**UNIVERSITY OF
WESTMINSTER** 

Table of Contents

Introduction	3
1 Literature Review	6
1.1 General Background of Mathematical Notetaking	6
1.2 Overview of Existing Applications	9
1.3 Large language models for mathematical notetaking	11
1.4 Risks, Sustainability, Accessibility & Ethics	13
2 Project Methodology	15
2.1 User Needs Analysis	15
2.1 Development Methodology	16
2.2. System Requirements	17
2.5 Data Sources and Tools	19
2.6 Evaluation Methodology	22
2.7 Ethical Considerations	23
3 Implementation	24
3.1 Architecture overview	24
3.1.1 Frontend	25
3.1.2 Backend	25
3.1.3 Database	26
3.1.4 Component interaction and data flow	27
3.1.5 Authentication and protected endpoints	27
3.2. Input methods	28
3.3 LaTeX generation process	29
3.4 UI/UX	30
4 Testing	35
4.1 Functional Testing (FT)	36
4.2 Validation Input Testing (VT)	37
4.3 Security Testing (ST)	39
4.4 Performance Testing (PT)	40
4.5 User Acceptance Testing (UAT)	41
5 Evaluation	44
6 Team Work and Collaboration	47
Conclusion	52
References	55
Supporting Materials and Appendices	60

Appendix 1: Prospective User Questionnaire.....	60
Appendix 2: Requirements Collection Spreadsheet.....	60
Appendix 3: Conference Poster and Abstract	60
Appendix 4: Testing and Evaluation Tables	60
Appendix 5: Workshop Presentation and Promotional Materials.....	61
Appendix 6: GitHub Repository.....	61

Introduction

The **Students-as-Co-Creators (SCC)** initiative at the University of Westminster promotes collaborative partnerships between students and academic staff in which students actively contribute to educational research, curriculum enhancement and the development of innovative learning technologies. Rather than acting solely as end users, students participate as researchers, designers and developers throughout the entire project lifecycle, from identifying educational challenges to evaluating and disseminating project outcomes. This collaborative approach enables students to develop research and professional skills while ensuring that educational technologies are designed around authentic student needs and experiences. The INTEGRAL project was undertaken as part of this initiative, bringing together staff and undergraduate students from the School of Computer Science and Engineering to investigate how recent advances in generative artificial intelligence can improve mathematical notetaking in higher education.

Mathematical notetaking is a fundamental component of learning within STEM disciplines. Research has shown that creating notes supports comprehension, problem solving and long-term knowledge retention. However, unlike conventional text, mathematical notation contains specialised symbols that remain difficult to produce efficiently using standard word-processing software. While LaTeX has become the de facto standard for producing professional mathematical documents, its syntax presents a substantial learning curve, requiring students to divide their attention between understanding mathematical concepts and learning formatting commands. This additional cognitive effort can negatively affect learning, particularly for students who are new to higher education or have additional learning needs.

Various digital solutions have attempted to address this challenge. Handwriting-based applications provide a natural means of entering mathematical expressions but typically depend on stylus-enabled tablets or touchscreen devices that are not universally available. Optical character recognition systems can convert handwritten equations into digital text, while recent large language models have demonstrated promising capabilities in understanding natural language descriptions of mathematics and generating accurate LaTeX representations. Despite these advances, existing tools often support only a single input modality, require specialised hardware, involve complex workflows or provide limited accessibility features. Consequently, there remains a need for an accessible, browser-based solution that enables students to create mathematical notes efficiently using commonly available university computers.

The **INTEGRAL (Interactive Notetaking Tool for Equations, Graphs, Algebra and Logic)** project addresses this challenge through the development of an AI-assisted web application capable of converting mathematical input into professionally formatted LaTeX in real time. The system supports two complementary methods of input: typed mathematical expressions and spoken mathematical language. These inputs are processed through a Python-based backend integrated with a large language model to generate structured LaTeX, which is immediately rendered within the browser. Unlike many existing mathematical editing tools, INTEGRAL requires no local installation and is specifically designed to operate within the University of Westminster computer laboratories, making the system accessible to students regardless of their personal hardware. The project, therefore, combines advances in artificial intelligence with principles of inclusive educational technology to improve mathematical note-taking for a diverse student population.

Beyond developing a functional prototype, the project investigates several broader educational questions. It explores how generative AI can improve mathematical notetaking practices, reduce cognitive load associated with mathematical formatting, improve

accessibility for students with different learning needs, and provide an equitable solution that does not depend on expensive specialist devices. In addition, the project considers the sustainability implications of AI-assisted digital notetaking compared with traditional paper-based approaches, together with the practical considerations of deploying such technology within institutional teaching environments. These aims closely reflect the original objectives defined in the SCC proposal and position the project as both a technical development and an educational research study.

The overall aim of the INTEGRAL project is to investigate how generative artificial intelligence can improve digital mathematical note-taking by developing and evaluating an accessible, browser-based application capable of converting mathematical expressions into correctly formatted LaTeX. The project combines software engineering with educational research to explore how AI can reduce the technical barriers associated with mathematical notation while preserving the cognitive processes that underpin mathematical learning. Emphasis is placed on usability, accessibility, privacy, institutional deployment and equitable access within higher education.

In accordance with the original Students-as-Co-Creators proposal, the project addresses the following research questions:

RQ1. How can generative AI improve mathematical note-taking practices?

RQ2. What specific challenges make digital mathematical note-taking difficult compared with handwritten methods?

RQ3. How can AI tools enhance the usability and accessibility of digital mathematical note-taking for students?

RQ4. In what ways can AI-assisted note-taking support learning, comprehension and retention in mathematics?

RQ5. What are the environmental and cognitive implications of paper-based versus AI-assisted note-taking?

RQ6. How can AI-assisted note-taking tools be designed to promote equitable access across different technological contexts?

To address these research questions, the project was organised around six objectives.

O1. Investigate the educational and technological context of digital mathematical note-taking. To establish the theoretical foundations of the project through a critical review of mathematical note-taking practices, digital mathematical input technologies, AI-assisted educational tools, accessibility, sustainability and ethical considerations, while identifying the limitations of existing approaches and the research gap addressed by INTEGRAL.

O2. Specify the system requirements and design the INTEGRAL architecture. To analyse user requirements and define the functional and non-functional requirements of the proposed system, together with an architecture capable of supporting secure, scalable and accessible AI-assisted mathematical note-taking within the University of Westminster environment.

O3. Design and implement the INTEGRAL application. To develop a browser-based mathematical note-taking application incorporating secure user authentication, note

management, mathematical editing, AI integration and a responsive user interface that supports efficient creation and management of mathematical notes.

O4. Develop and integrate an AI-assisted mathematical conversion pipeline. To implement and evaluate an AI-assisted processing pipeline capable of converting typed and spoken mathematical expressions into structured LaTeX using locally deployed language models, while maintaining low latency, data privacy and compatibility with institutional computing facilities.

O5. Test and evaluate the developed system. To verify that the completed prototype satisfies its functional and non-functional requirements through comprehensive functional, validation, security, compatibility, integration, performance, usability and user acceptance testing, together with analysis of the quality and reliability of the generated mathematical output.

O6. Evaluate the educational contribution and future potential of the project. To critically evaluate the accessibility, usability, educational value, ethical implications, sustainability, limitations and future development of AI-assisted mathematical note-taking, while reflecting on the Students-as-Co-Creators partnership and its contribution to the successful delivery of the project.

The principal outputs of the project are summarised by the following deliverables.

D1. Literature Review and Problem Analysis. A comprehensive review of mathematical note-taking, digital mathematical input technologies, generative AI, accessibility, sustainability and educational theory, establishing the motivation and research gap for the project.

D2. Requirements Specification and System Design. A complete specification of the functional and non-functional requirements together with the overall architecture, workflows and technical design of the INTEGRAL system.

D3. INTEGRAL Web Application. A browser-based mathematical note-taking application supporting secure user authentication, note management, mathematical editing, AI-assisted LaTeX generation and responsive interaction through an intuitive user interface.

D4. AI-Assisted Mathematical Processing Pipeline. An integrated processing pipeline supporting text-to-LaTeX and speech-to-LaTeX conversion using locally deployed AI models, together with the supporting backend services required for secure and efficient operation.

D5. Testing and Evaluation. A comprehensive evaluation of the developed system, including functional, validation, security, compatibility, integration, performance, usability and user acceptance testing, together with analysis of system performance, limitations and reliability.

D6. Final Project Report and Recommendations. A complete project report documenting the research background, methodology, implementation, testing, evaluation, Students-as-Co-Creators collaboration, conclusions and recommendations for future research and continued development.

The relationships between the research questions, objectives and deliverables are illustrated throughout the remainder of this report. Chapter 2 establishes the theoretical background and addresses from **O1** through to **D1**. Chapter 3 presents the research methodology, requirements analysis and system design corresponding to **O2** and **D2**. Chapter 4 describes the implementation of the INTEGRAL application and AI processing pipeline,

addressing **O3** and **O4** through to **D3** and **D4**. Chapter 5 presents the testing strategy and evaluation results associated with **O5** and **D5**. Chapter 6 evaluates the completed prototype, discusses its educational contribution, reflects on the Students-as-Co-Creators partnership, and considers future work, thereby addressing **O6** and **D6**. Finally, the concluding chapter summarises the principal contributions of the project and assesses the extent to which the original research aim and research questions have been addressed.

1 Literature Review

1.1 General Background of Mathematical Notetaking

Note-taking is often framed as a traditional or even outdated study practice, frequently associated with relatively passive models of learning (Friedman, 2014). However, this characterisation is increasingly difficult to sustain considering constructivist theories of pedagogy (Castello and Monereo, 2005), which position learning as an active process of making meaning rather than receiving information (Biggs, Tang, and Kennedy, 2022). From this perspective, effective notetaking is not merely the recording of knowledge, but its active construction wherein learners selectively attend to, interpret, reorganise, and encode information in ways that reflect their prior understanding and evolving cognitive framing (Castello and Monereo, 2005). It therefore remains central to contemporary pedagogy precisely because it externalises thinking, supports reflection, and produces artefacts that can be iteratively revisited and refined into more sophisticated conceptual structures (Friedman, 2014).

In mathematics education, note-taking plays a particularly significant role because students must capture not only written explanations but also symbolic notation, logical arguments, diagrams, and worked examples that collectively encode mathematical reasoning processes (Schoenfeld, 2014; Duval 2006). Consequently, mathematical notes function not simply as records of content, but as representations of conceptual structure and procedural thinking that support subsequent problem-solving, revision, and abstraction (Hiebert and Lefevre, 2013). In this context, the quality, accessibility, and expressive flexibility of mathematical notation directly influence how effectively learners engage with complex ideas (Arcavi, 1994).

The widespread adoption of digital technologies in higher education has significantly reshaped note-taking in practice (Friedman, 2014). Importantly, this shift from handwritten to digital environments does not simply constitute a change of medium; rather, it expands the representational and epistemic possibilities available to learners (Bolter, 2001). Notes are no longer necessarily static artefacts, but can instead become dynamic, networked, and computational objects: computational notebook environments now allow code, explanation, and output to sit within a single document (Kluyver et al., 2016), while AI-supported systems increasingly mediate how learners search for, question, and synthesise information as they take notes (Holmes, Bialik and Fadel, 2019). In many respects, this development makes note-taking more fluid and more closely aligned with the epistemic practices of computationally oriented disciplines such as computer science (Kluyver et al., 2016).

However, these gains are not without complications. New frictions have emerged where domain-specific notation systems lag behind the ease of natural text entry (Norman, 2013). This challenge is particularly acute within mathematics because mathematical notation

contains specialised symbolic structures, including fractions, matrices, superscripts, integrals, and formal operators that typically require specialised input methods beyond conventional text entry (Duval, 2006; Arcavi, 1994). The central difficulty is therefore not mathematical problem-solving itself, but mathematical representation: students may understand a concept while simultaneously struggling to express it efficiently within digital environments (Duval, 2006).

The most widely adopted solution to digital mathematical writing is LaTeX, which has become the international standard for mathematical and scientific typesetting (Lamport, 1994; Gkritzali *et al.*, 2024). Its expressive precision makes it indispensable within academic publishing and professional communication (Lamport, 1994). However, an important distinction exists between typesetting mathematics and learning mathematics: LaTeX's dominance as a publishing standard says little about its suitability as a real-time learning or note-taking tool (Gkritzali *et al.*, 2024). LaTeX was designed to produce print-ready documents rather than to support rapid note-taking during lectures or seminars (Lamport, 1994). Before students can record mathematical ideas fluently, they must first learn a specialised markup language composed of commands and syntax that are largely unrelated to the underlying mathematical concepts themselves (Gkritzali *et al.*, 2024). Although such knowledge may ultimately become professionally valuable, it nevertheless introduces an additional technical learning task that does not directly contribute to conceptual mathematical understanding (Gkritzali *et al.*, 2024).

This distinction highlights a broader pedagogical tension. Students should ideally devote cognitive resources towards abstraction, pattern recognition, and mathematical reasoning, yet in practice they may instead divide attention between understanding lecture content and managing notation syntax or complex equation editors (Sweller, 1988). Cognitive load theory provides a useful analytical lens in this regard by distinguishing between intrinsic load (the inherent complexity of the material), germane load (the cognitive effort devoted to schema construction), and extraneous load (the load imposed by suboptimal instructional design or tools), although more recent formulations treat these categories as interacting rather than strictly additive (Sweller, 1988; Sweller, van Merriënboer and Paas, 1998). Digital mathematical tooling therefore cannot be understood as pedagogically neutral, since it actively redistributes cognitive effort across these categories (Mayer and Moreno, 2003). In many educational contexts, seminar design attempts to minimise extraneous load through restrictive or simplified tooling choices (Sweller, van Merriënboer and Paas, 1998). However, this may inadvertently limit the richer forms of computational engagement that contemporary digital environments could otherwise support (Bolter, 2001).

Artificial intelligence further complicates this landscape by reshaping how mathematical learning is mediated and by raising substantive questions regarding its compatibility with established theories of learning (Holmes, Bialik and Fadel, 2019). From a constructivist standpoint, learning depends fundamentally upon the active generation, manipulation, and restructuring of knowledge rather than the passive reception in its implications (Martin *et al.*, 2025; Chi and Wylie, 2014; Wittrock 1992). While AI may reduce extraneous load by handling syntactic complexity and representation management, it may also unintentionally reduce germane load if learners bypass the effortful processes required for schema construction (Martin *et al.*, 2025). Consequently, AI should not be understood as a neutral addition to the learning environment, but rather as a force that actively redistributes cognitive work, necessitating careful consideration of when it enhances learning and when it risks displacing it (Martin *et al.*, 2025; Holmes, Bialik and Fadel, 2019). AI systems capable of generating

proofs, completing derivations, or translating informal prompts into formal mathematical notation therefore occupy an ambiguous pedagogical position: they may function as scaffolds that reduce representational friction, but equally risk collapsing the productive struggle through which conceptual understanding is often developed (Kapur, 2014; Dilling and Herrmann, 2024). Cognitive load theory is similarly ambivalent in its implications (Martin *et al.*, 2025). While AI may reduce extraneous load by handling syntactic complexity and representation management, it may also unintentionally reduce germane load if learners bypass the effortful processes required for schema construction (Martin *et al.*, 2025). Consequently, AI should not be understood as a neutral addition to the learning environment, but rather as a force that actively redistributes cognitive work, necessitating careful consideration of when it enhances learning and when it risks displacing it (Martin *et al.*, 2025; Holmes, Bialik and Fadel, 2019).

Equity and accessibility introduce an additional and often under-emphasised layer of constraint in the design of digital mathematical learning environments (CAST, 2024). In institutional contexts, accessibility is frequently operationalised through compliance frameworks such as the Blackboard Ally accessibility score (now rebranded Anthology Ally), which evaluates the machine-readability and assistive-technology compatibility of learning materials (Anthology, 2026). However, mathematical notation poses persistent challenges in this regard: expressions rendered as images, poorly structured HTML, or inaccessible markup may significantly reduce accessibility for screen-reader users and other assistive technologies, and this includes LaTeX source itself, which remains difficult for many screen readers to narrate accurately (Manzoor *et al.*, 2019; Maćkowski *et al.*, 2023). Yet equity considerations extend beyond formal compliance metrics (CAST, 2024). They also encompass broader questions of cognitive and technological access, particularly given that students vary substantially in prior exposure to LaTeX, equation editors, and structured digital authoring systems (Gkritzali *et al.*, 2024). As a result, representational choices in mathematical communication can systematically advantage or disadvantage particular learner groups, especially where expressive tooling requires substantial upfront technical investment (Maćkowski *et al.*, 2023). Accessibility should therefore be understood not merely as inclusion for disabled learners, but more fundamentally as equitable access to the cognitive and technical infrastructures through which mathematical ideas are expressed and developed (CAST, 2024).

These considerations reveal a research gap at the intersection of mathematical cognition, digital tooling, and artificial intelligence. Existing approaches tend broadly to fall into two unsatisfactory categories: systems that support expressive mathematical notation while imposing substantial usability and cognitive barriers (Manzoor *et al.*, 2019), and general-purpose AI tools that reduce friction but fail to preserve the formal structure and epistemic precision required for mathematical thinking (Dilling and Herrmann, 2024). This bifurcation leaves a substantial design space underexplored, namely systems capable of reducing barriers to digital mathematical expression without displacing the cognitive processes that underpin learning. There is a need for tools that integrate AI in ways that are transparent, pedagogically grounded, and explicitly sensitive to cognitive load while remaining accessible across diverse institutional and technological contexts (Martin *et al.*, 2025; CAST, 2024). Addressing this gap would not only advance the design of educational technologies but also provide a concrete mechanism for aligning contemporary AI capabilities with principles of equitable, rigorous, and cognitively coherent mathematical learning.

1.2 Overview of Existing Applications

Existing digital tools for mathematical typesetting are predominantly designed for professional and academic users, although a smaller but growing subset is explicitly oriented towards educational use. This creates a variable but persistent misalignment between the design priorities of many tools and the cognitive and pedagogical requirements of learning mathematics. In practice, many existing systems solve only a single aspect of the broader problem of digital mathematical note-taking while leaving other challenges unresolved. Collaborative LaTeX editors support high-quality formatting and publication workflows; symbol-recognition systems simplify notation entry; and AI-driven tools provide increasingly sophisticated explanation and generation capabilities. However, these capabilities are not consistently calibrated to the practical realities of students taking notes during lectures or seminars, nor to the cognitive processes through which mathematical understanding is developed (Mueller and Oppenheimer, 2014). With the former, digital mathematical note-taking can become a secondary technical task competing with rather than supporting conceptual learning, while with the latter, systems that generate complete answers or derivations may reduce the generative cognitive effort required of learners, increasing the risk of cognitive offloading rather than supporting active engagement with mathematical ideas.

Within this landscape, collaborative authoring environments such as Overleaf exemplify a production-oriented design paradigm (Sanches Egea et al., 2026; Bahls and Wray, 2015). Overleaf provides a highly capable cloud-based interface for LaTeX authoring and is widely adopted within scientific and technical disciplines because it enables precise mathematical typesetting and collaborative document preparation for reports, dissertations, and research publications. However, its pedagogical suitability for novice learners remains questionable. LaTeX introduces substantial representational overhead, requiring learners to acquire procedural fluency in a specialised markup language that is often orthogonal to the intended learning objectives (Bahls and Wray, 2015; Knauff and Nejasmic, 2014; Gratzner, 2008). While this overhead may ultimately contribute to disciplinary enculturation and professional practice, it can also impose significant extraneous cognitive load, particularly in time-constrained educational settings such as lectures and seminars (Sweller, 1988; Quinby et al., 2020). Students may therefore find themselves dividing attention between understanding mathematical concepts and recalling syntactic commands or formatting conventions. Consequently, Overleaf is highly effective as a publication-oriented environment, but less clearly aligned with equitable, cognitively accessible, and efficient mathematical note-taking during learning activities.

A contrasting design approach is illustrated by systems such as Detexify, which address the narrower but important problem of mathematical symbol input through handwritten gesture recognition (Kirsch, 2010). Detexify is built around k-nearest neighbour pattern matching enabling the classification of handwritten mathematical symbols into corresponding LaTeX commands. To a contemporary reader, it may be somewhat surprising that Detexify is not supported by a CNN architecture to learn spatial hierarchies in visual input, reflecting the rapid rise of AI since Detexify emerged in 2010. Irrespective of the enabling architecture, the Detexify system reduces a specific form of representational friction, namely symbol lookup and notation entry, while also reducing the need for users to memorise large numbers of LaTeX commands. This makes mathematical input more intuitive and accessible, particularly for unfamiliar notation. Nevertheless, the pedagogical scope of the system remains limited

primarily to interface efficiency rather than broader support for mathematical reasoning or conceptual learning. Users must still understand LaTeX syntax and manually construct complete mathematical expressions, limiting the system's effectiveness as a comprehensive educational note-taking environment.

More recent AI systems, including tools such as Perplexity AI and ChatGPT, illustrate the rapid integration of large language models (LLMs) into both knowledge work and educational contexts (Kasneci et al., 2023). Unlike systems based on statistical learning such as Detexify, which operate through localised visual feature extraction and pattern recognition, contemporary LLMs are transformer-based architectures trained on large-scale corpora to model long-range linguistic dependencies through self-attention mechanisms (Vaswani et al., 2017). This enables flexible natural-language interaction, contextual summarisation, mathematical explanation generation, and increasingly sophisticated formatting capabilities. Consequently, LLMs address a substantially broader interactional problem than notation recognition alone, reducing friction by allowing users to express intent conversationally rather than through formal symbolic syntax.

However, these same capabilities introduce important pedagogical tensions. While AI systems can make mathematical information more accessible and reduce the technical effort associated with notation production, their primary function is often to explain or solve mathematical problems rather than to support students in recording and developing their own understanding. This distinction is educationally significant because mathematical learning depends not only on exposure to correct solutions, but on the active generation, manipulation, and externalisation of reasoning processes. By generating intermediate steps or complete derivations, such systems may reduce the generative cognitive effort required of learners, thereby increasing the risk of cognitive offloading rather than supporting active mathematical engagement (Risko and Gilbert, 2016; Sparrow et al., 2011). In this respect, AI systems risk functioning not as supports for learning, but as substitutes for important forms of conceptual and procedural activity.

Taken together, these systems reveal a fragmented design space shaped by three partially competing objectives: expressive mathematical precision, low-friction interaction, and pedagogically meaningful cognitive engagement. Learning-based tools such as Detexify reduce representational barriers at the level of symbol input but remain narrow in scope; Overleaf provides expressive power at the cost of accessibility and cognitive overhead; and LLM-driven tools offer unprecedented flexibility while introducing concerns around epistemic reliability, learner dependency, and the displacement of active reasoning processes. While each approach addresses part of the problem, none fully integrates intuitive mathematical input, accessible digital note-taking, and responsible AI assistance within a single educational system. Despite rapid technological progress across each of these directions, comparatively little work has addressed how they might be synthesised into an equitable note-taking environment that is simultaneously accessible, cognitively appropriate, pedagogically grounded, and capable of preserving the generative nature of mathematical learning in authentic educational contexts (Wang et al., 2024). This unresolved intersection constitutes the central research gap addressed by the present work.

1.3 Large language models for mathematical notetaking

Large Language Models (LLMs) constitute a prominent class of contemporary AI systems concerned with generating and processing natural language, and are situated within a broader AI landscape encompassing rule-based systems, statistical machine learning, computer vision, and deep neural architectures (Russell and Norvig, 2021; Goodfellow et al, 2016). Modern LLMs are typically implemented using transformer architectures, in which self-attention mechanisms enable the modelling of long-range dependencies across sequences of tokens (Vaswani et al., 2017; Devlin et al., 2019). Trained on large-scale corpora, these models learn distributed statistical representations of language that support a wide range of downstream tasks, including question answering, summarisation, code generation, translation, and increasingly sophisticated mathematical reasoning (Brown et al., 2020; OpenAI, 2023; Wei et al., 2022). Public-facing systems such as ChatGPT have accelerated the adoption of LLMs across educational and professional domains, reshaping expectations around information access, interaction paradigms, and digital productivity.

From a systems perspective, LLM capabilities may be accessed either through hosted conversational interfaces or via application programming interfaces (APIs), which enable integration into domain-specific software environments. This distinction is particularly significant for educational technology design, as API-based deployment allows LLM functionality to be embedded within bespoke pedagogical tools rather than operating solely as general-purpose chat systems. In the context of mathematical education, this enables the development of systems that support translation between informal learner input and formal symbolic representations, thereby functioning as a representational mediation layer between natural language and structured mathematical notation. Within the present work, this mediation role is deliberately constrained: the LLM is intended to reduce representational friction in mathematical note-taking rather than to generate solutions or replace learner reasoning. In this sense, the system is designed to preserve the generative cognitive processes central to mathematical learning while reducing the technical overhead associated with formal expression.

For mathematical note-taking, this capability is particularly relevant because students are often required to translate rapidly between conceptual understanding and formal symbolic notation. Rather than focusing on solving problems, the primary educational challenge in many learning contexts lies in efficiently recording mathematical ideas using systems such as LaTeX or equation editors. LLMs can support this process by converting natural language, spoken input, or informal mathematical descriptions into structured markup. This reduces dependence on memorising syntactic commands and navigating complex interface workflows, thereby lowering technical barriers that may otherwise interfere with conceptual engagement. Importantly, this distinction between mathematical representation and mathematical reasoning is pedagogically significant: effective learning depends not only on producing correct outputs but on actively engaging with the processes of abstraction, reasoning, and formalisation.

At the same time, LLMs are probabilistic sequence models rather than formally verified reasoning systems and therefore introduce important limitations in reliability and correctness (Bender et al., 2021; Ji et al., 2023). They may produce hallucinated, inconsistent, or mathematically incorrect outputs, meaning that generated content cannot be treated as inherently authoritative. Within educational contexts, this necessitates transparency and

opportunities for user verification and correction, ensuring that AI outputs remain subject to critical engagement rather than passive acceptance. Although much LLM research evaluates performance using benchmarks such as MMLU, GSM8K, or HumanEval, alongside metrics such as exact-match accuracy, semantic similarity, and human evaluation, pedagogical suitability depends not only on correctness but also on interpretability, consistency, and alignment with learning objectives (Hendrycks et al., 2021; Cobbe et al., 2021; Chen et al., 2021). Consequently, evaluation in educational settings must extend beyond technical performance to include learning impact and cognitive consequences.

The integration of LLMs into educational systems also raises broader concerns relating to data privacy, institutional governance, reproducibility, and sustainability (Bommasani et al., 2021; Weidinger et al., 2022; Patterson et al., 2021). Cloud-based LLM services typically process user inputs on external infrastructure, creating potential risks associated with the handling of student-generated data. In response, locally deployed and open-source models have emerged as an alternative deployment strategy, enabling institutions to retain control over data flows, improve auditability, and reduce dependence on proprietary ecosystems. While local deployment may involve trade-offs in raw performance compared to large commercial systems, it can offer advantages in transparency, adaptability, and long-term sustainability. These considerations also extend to energy consumption and computational cost, as large-scale transformer inference can impose significant environmental and infrastructural burdens. From an equity perspective, reliance on proprietary AI services may also introduce financial and institutional barriers that unevenly shape access to advanced educational technologies, whereas open-source and locally deployable systems can support more equitable and reproducible deployment across diverse contexts.

Taken together, LLM-based educational tools reveal a fragmented design space shaped by multiple competing objectives. On one hand, these systems offer powerful mechanisms for reducing technical friction in mathematical notation, improving accessibility, and supporting flexible interaction through natural language interfaces. On the other hand, they risk undermining the cognitive processes that underpin mathematical learning if they are used to replace rather than support student reasoning. The central pedagogical tension lies in ensuring that AI reduces extraneous technical load without diminishing the generative cognitive effort required for conceptual development. Despite rapid advances in both commercial and open-source systems, comparatively little work has focused on LLMs as assistive technologies for mathematical note-taking specifically, that is, systems designed to support students in recording and representing their own understanding during learning activities rather than solving problems on their behalf.

This unresolved intersection constitutes a significant research gap. There remains a need for educationally grounded AI systems that integrate LLMs in ways that are transparent, cognitively appropriate, and aligned with active learning principles, while simultaneously addressing practical constraints of usability, accessibility, privacy, and institutional deployment. Such systems should preserve student ownership of mathematical reasoning while reducing the representational and technical barriers associated with formal notation, thereby enabling more effective, equitable, and cognitively coherent mathematical learning environments.

1.4 Risks, Sustainability, Accessibility & Ethics

The integration of Artificial Intelligence (AI) into educational technology presents significant opportunities to improve accessibility, reduce technical barriers to learning, and support more flexible forms of mathematical engagement. However, it simultaneously raises complex ethical, pedagogical, and socio-technical questions relating to cognitive integrity, accessibility, data governance, and sustainability. These considerations are not independent; rather, they form an interconnected set of constraints that shape the design and deployment of AI-assisted educational systems.

Accessibility and educational equity introduce an additional layer of constraint. AI-based systems can improve accessibility by reducing reliance on specialised technical knowledge, supporting multiple modes of expression, and offering adaptive forms of assistance tailored to different levels of prior understanding. However, these benefits are unevenly distributed when access depends on proprietary platforms, commercial subscriptions, or high-performance hardware. This raises concerns of infrastructural inequality, particularly where students have differing exposure to digital mathematical systems or assistive technologies.

Accessibility must also be understood in both specific and general terms. On one hand, mathematical notation presents challenges for assistive technologies such as screen readers, especially when expressions are encoded as images or poorly structured markup. Standards such as LaTeX and MathML can improve machine readability, but they often require specialist knowledge that exceeds what is typically expected of students. On the other hand, accessibility also refers more broadly to reducing unnecessary technical barriers for all learners, ensuring that students can express mathematical ideas effectively in digital environments without excessive syntactic or interface burden. In this sense, accessibility is not solely a disability inclusion concern, but a universal design principle related to cognitive and technical usability.

In addition, the use of locally deployed models introduces a significant shift in data governance and institutional control. Unlike cloud-based systems, where student-generated content is transmitted to external servers for processing, local inference ensures that all user data remains within the institutional environment. This is particularly important in educational contexts where mathematical notes may constitute sensitive or identifiable student work, bringing the system into scope of data protection frameworks such as the General Data Protection Regulation (GDPR). Beyond compliance, this architectural choice reflects a broader principle of data sovereignty, whereby educational institutions retain control over the storage, processing, and lifecycle of student data rather than delegating these responsibilities to third-party providers.

Sustainability introduces a further dimension of analysis. Training and deploying large transformer-based models is computationally and energy intensive, requiring substantial hardware infrastructure and ongoing inference costs [reference]. While individual model queries may appear negligible, aggregate usage across large student populations can result in significant energy consumption. In this sense, environmental impact must be understood as a systemic property of scale rather than an individual interaction cost. Traditional handwritten note-taking provides a near-zero operational energy baseline, although it lacks the representational flexibility and computational augmentation of digital systems. This

comparison highlights a fundamental trade-off between computational capability and environmental cost in AI-mediated education.

However, sustainability is not solely a constraint but also a design parameter. Deployment decisions such as model size, architecture, and hosting strategy can be shaped to optimise efficiency. Techniques such as model distillation and quantisation, as well as the use of smaller task-specific open-source models, can significantly reduce computational overhead while maintaining adequate performance for educational tasks such as natural-language-to-markup translation. Local deployment further reduces dependence on large-scale cloud infrastructure and repeated network communication, offering potential improvements in energy efficiency alongside enhanced institutional control.

A further practical consideration concerns the dependence of cloud-based LLMs on internet connectivity and external infrastructure. In lecture and seminar environments, where students interact with mathematical ideas in real time, network latency, bandwidth fluctuations, and service availability can introduce unpredictable delays that disrupt cognitive continuity and the flow of learning. Locally deployed models avoid these infrastructural dependencies, providing consistent, low-latency performance that supports responsive interaction and ensures that system reliability is determined by pedagogical rather than network constraints.

The selection of model scale introduces a fundamental trade-off between computational efficiency and representational capability. Smaller models (e.g., 7B parameter systems) typically offer faster inference and lower memory requirements, making them more suitable for real-time educational applications where responsiveness is critical. Larger models (e.g., 34B or 70B parameter systems), while potentially offering improved reasoning and generation quality, introduce significantly higher computational overhead and latency, which may exceed acceptable thresholds for interactive classroom use. This trade-off highlights the importance of aligning model architecture with pedagogical constraints rather than purely optimising for benchmark performance.

A further dimension of local deployment concerns the removal of per-request API costs associated with commercial LLM services. While often framed as a technical implementation detail, this has broader implications for the scalability and long-term sustainability of educational tools. Eliminating dependency on usage-based pricing models reduces financial barriers to adoption and ensures that system access is not constrained by institutional subscription budgets or fluctuating provider pricing structures. In this sense, cost efficiency becomes a structural factor in ensuring equitable and scalable deployment of AI-assisted educational technologies.

Taken together, these considerations suggest that AI-assisted educational tools must be evaluated within a broader socio-technical framework that integrates cognitive, ethical, and environmental dimensions. Effective design requires balancing three interdependent constraints: preserving cognitive integrity and generative engagement in learning; ensuring equitable and accessible participation across diverse learner populations and institutional contexts; and minimising environmental and infrastructural cost while maintaining functional performance. Rather than treating these as separate design requirements, they should be understood as co-equal and interdependent factors shaping the responsible development of AI-mediated mathematical learning systems.

2 Project Methodology

This chapter describes the methodology adopted throughout the INTEGRAL project, from identifying user needs and eliciting system requirements to the design, implementation, and evaluation of the proposed AI-assisted mathematical note-taking application. A user-centred and iterative software development approach was employed, combining requirements gathered from prospective users with findings from the literature review to guide the system design. The chapter describes the requirements elicitation process, the development methodology, the specification of functional and non-functional requirements, the tools and technologies used during implementation, the evaluation strategy, and the ethical considerations that informed the project's design and deployment. This chapter describes the methodology adopted throughout the INTEGRAL project, from identifying user needs and eliciting system requirements to the design, implementation, and evaluation of the proposed AI-assisted mathematical note-taking application. A user-centred and iterative software development approach was employed, combining requirements gathered from prospective users with findings from the literature review to guide the system design. The chapter describes the requirements elicitation process, the development methodology, the specification of functional and non-functional requirements, the tools and technologies used during implementation, the evaluation strategy, and the ethical considerations that informed the project's design and deployment.

2.1 User Needs Analysis

The development of INTEGRAL followed a user-centred design approach in which prospective users were involved at the earliest stages of the project to identify the challenges associated with digital mathematical note-taking and to establish the requirements for the proposed system. Rather than defining functionality solely from a technical perspective, the project sought to understand the needs of students and staff who regularly work with mathematical notation in higher education.

To support this process, a structured questionnaire was designed and distributed to prospective users (see **Appendix 1**). The questionnaire investigated participants' current note-taking practices, the difficulties they encounter when recording mathematical content, their experience with LaTeX, accessibility requirements, preferred methods of mathematical input, and the features they considered most valuable in an AI-assisted mathematical note-taking application. The complete questionnaire is provided in **Appendix 1**.

The questionnaire comprised twelve questions organised into five themes:

- participant background and current note-taking practices;
- challenges associated with mathematical notation and digital note-taking;
- accessibility and inclusive design requirements;
- desired functional features and system capabilities; and
- intended use of the proposed system within teaching and learning activities.

Collecting these requirements before implementation ensured that the system addressed genuine educational needs rather than technology-driven assumptions. The findings informed

both the functional and non-functional requirements presented in the following section and guided subsequent architectural and interface design decisions.

The questionnaire findings were combined with evidence from the literature review and discussions within the Students-as-Co-Creators project team to produce a structured requirements specification. Rather than documenting requirements informally, all identified requirements were recorded within a dedicated requirements management spreadsheet (see **Appendix 2**), enabling systematic classification, prioritisation, traceability, and validation throughout the development process.

Each requirement was documented using a standard template consisting of:

- requirement identifier;
- requirement type (functional or non-functional);
- description;
- user story;
- acceptance criteria using the **Given–When–Then** format;
- implementation priority following the **MoSCoW** prioritisation technique;
- quality attribute (where applicable);
- measurable success criteria;
- affected system component; and
- implementation status.

Functional requirements described the behaviour expected from the application, including AI-assisted LaTeX generation, speech input, note management, authentication, export functionality, and user interface features. Non-functional requirements captured quality attributes such as accessibility, performance, security, usability, reliability, maintainability, and privacy. Examples include target response latency, WCAG accessibility compliance, HTTPS communication, user data isolation, and Blackboard Ally compatibility. These requirements provided measurable objectives against which the completed prototype could subsequently be evaluated.

2.1 Development Methodology

The development of INTEGRAL followed an **iterative prototyping methodology**, enabling the system to be designed, implemented, evaluated, and refined through successive development cycles. This approach was selected because the project combined software engineering with emerging artificial intelligence technologies, requiring continuous experimentation, validation, and refinement throughout development (Sommerville, 2020; Pressman and Maxim, 2020).

The development process began with the identification of user needs and the specification of functional and non-functional requirements. These requirements provided the basis for the initial system architecture and guided the prioritisation of implementation tasks. Rather than attempting to develop the complete application in a single stage, individual components were implemented incrementally, allowing each subsystem to be tested and evaluated before integration into the overall system (Pressman and Maxim, 2020).

The project progressed through a series of development iterations that included user interface design, backend implementation, AI model integration, speech-to-text processing, mathematical notation generation, authentication, note management, and system optimisation. Following each iteration, the prototype was evaluated against the specified requirements, with the findings informing subsequent design improvements and

implementation decisions. This iterative approach is consistent with user-centred software development practices, where continuous evaluation and refinement improve both system quality and usability (Norman, 2013; ISO 9241-210, 2019).

Throughout development, continuous testing was performed to verify functionality, usability, security, performance, and compatibility. Feedback obtained from prototype demonstrations and internal project reviews was used to refine both the system architecture and the user interface. This iterative process reduced implementation risks, facilitated the integration of new features, and ensured that the final prototype remained aligned with the educational objectives and user requirements established during the earlier stages of the project (Sommerville, 2020).

The overall development lifecycle adopted within the project is illustrated in **Figure 2.1**.

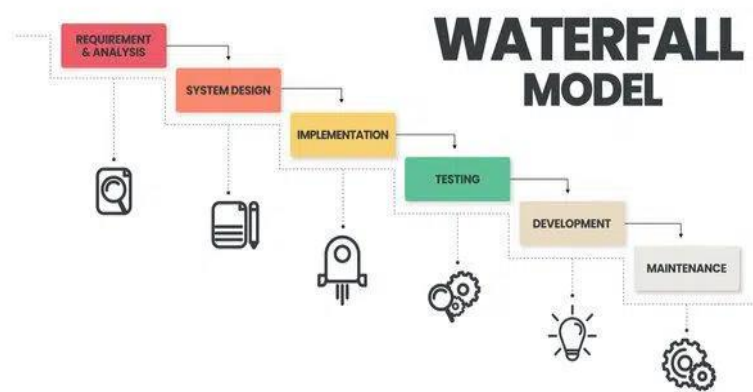


Fig. 2.1 Project Development Model

2.2. System Requirements

The information gathered during the requirements elicitation process was translated into a structured Software Requirements Specification (SRS), providing a clear definition of the functionality and quality attributes expected from the INTEGRAL system. Following established software engineering practice, requirements were categorised as functional and non-functional to ensure that both the system behaviour and its performance characteristics could be systematically designed, implemented, and evaluated (Sommerville, 2020; Pressman and Maxim, 2020). The complete requirements specification, including user stories, acceptance criteria, priorities, implementation status, and traceability information, is presented in **Appendix X**.

The functional requirements describe the services that the application must provide to support AI-assisted mathematical note-taking. These include user authentication, note creation and management, speech recognition, natural language processing, mathematical notation generation, file management, and personalisation features. The requirements were derived from the user needs analysis presented in Section 2.1 and were refined through discussions within the Students-as-Co-Creators project team. Collectively, they formed the basis for the overall system architecture and guided the implementation of individual software components.

Figure 2.2 illustrates the workflow adopted for AI-assisted mathematical notation generation. User input is validated before being processed by a locally deployed Mistral 7B language model via the Ollama inference framework. The generated LaTeX output is subsequently post-processed and returned to the frontend through the system's API. The inclusion of input validation and response caching was specified as part of the functional requirements to improve reliability and minimise unnecessary model invocations.

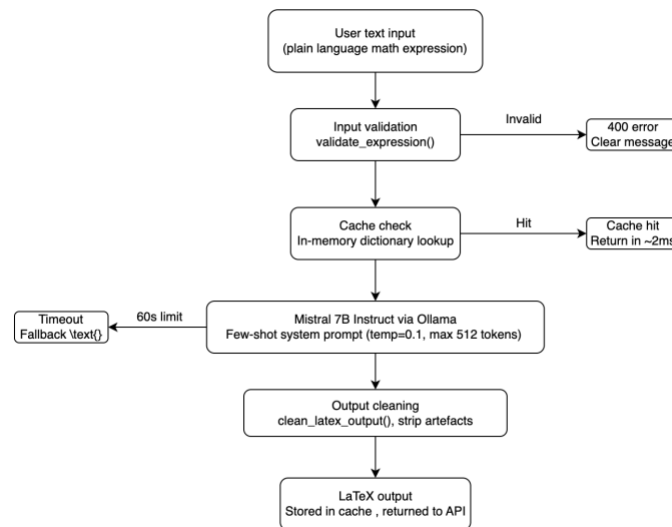


Figure 2.2; Text-to-LaTeX Generation Flowchart

To support hands-free note-taking, speech recognition functionality was incorporated into the system requirements. As illustrated in Figure 2.3, spoken mathematical expressions are first transcribed using the Whisper speech recognition model before entering the same LaTeX generation pipeline. Reusing the downstream processing workflow provides consistent behaviour regardless of whether mathematical expressions are entered through text or speech while reducing implementation complexity.

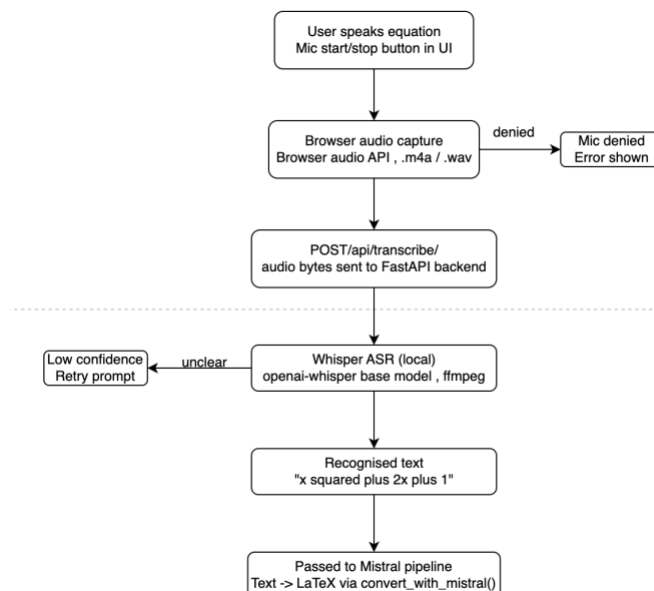


Figure 2.3 Speech-to-text processing workflow

In addition to functional capabilities, the project defined a comprehensive set of non-functional requirements covering performance, accessibility, security, maintainability, scalability, and privacy. These quality attributes were particularly important because the application is intended for use during live lectures, where rapid response times and reliable mathematical transcription are essential. The decision to deploy all AI models locally was therefore driven not only by performance considerations but also by the need to preserve user privacy and reduce dependence on external cloud services.

Several open-source language models were evaluated during the system design process to determine their suitability for local deployment. As shown in Figure 2.4, candidate models were compared with respect to reasoning capability, mathematical performance, computational efficiency, and hardware requirements. Based on this evaluation, Mistral 7B was selected as it provided the most appropriate balance between inference quality, response time, and resource consumption, satisfying the project's non-functional requirements while remaining deployable on commodity hardware.

Model	Modality	MMLU	HellaSwag	WinoGrande	PIQA	Arc-e	Arc-c	NQ	TriviaQA	HumanEval	MBPP	MATH	GSM8K
LLaMA 2 7B	Pretrained	44.4%	77.1%	69.5%	77.9%	68.7%	43.2%	24.7%	63.8%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	Pretrained	55.6%	80.7%	72.9%	80.8%	75.2%	48.8%	29.0%	69.6%	18.9%	35.4%	6.0%	34.3%
Code LLaMA 7B	Finetuned	36.9%	62.9%	62.3%	72.8%	59.4%	34.5%	11.0%	34.9%	31.1%	52.5%	5.2%	20.8%
Mistral 7B	Pretrained	60.1%	81.3%	75.3%	83.0%	80.0%	55.5%	28.8%	69.9%	30.5%	47.5%	13.1%	52.1%

Figure 2.3 Benchmark Evaluations of LLM's

The resulting requirements specification served as the primary reference throughout development, ensuring traceability between identified user needs, implemented functionality, and the evaluation criteria used to assess the completed prototype.

2.5 Data Sources and Tools

The development of INTEGRAL relied on a combination of primary and secondary data sources together with a range of software tools supporting system design, implementation, testing, and evaluation. Using multiple sources of evidence and specialised development tools ensured that both the educational objectives and technical requirements of the project could be addressed effectively.

The primary data source comprised the prospective user questionnaire described in Section 2.1. Responses from students and staff were analysed to identify common challenges associated with mathematical note-taking, accessibility requirements, and desired system functionality. These findings informed the elicitation and prioritisation of the system requirements and guided design decisions throughout the development process.

Secondary data sources consisted of academic literature, technical documentation, and software engineering standards. The literature review provided the theoretical foundation for the project by examining digital note-taking, mathematical notation systems, accessibility in higher education, speech recognition technologies, and generative artificial intelligence. Official documentation for the selected development frameworks and AI models was also consulted throughout implementation to ensure that best practices were followed.

A range of software tools and frameworks was used to implement the system. The user interface was developed using React and TypeScript, which provide the components and programming structure required to create an interactive web application. The backend

services were developed in Python using the FastAPI framework, which manages communication between the user interface, application logic, and data storage components.

Artificial intelligence functionality was integrated through Ollama, which enables local execution of the Mistral 7B large language model. This model provides natural language processing capabilities, allowing the system to interpret user input and generate mathematical content, including LaTeX representations of equations. Speech recognition was implemented using Whisper, which converts spoken mathematical expressions into text that can then be processed by the application. Mathematical notation was displayed using KaTeX, a rendering library that converts LaTeX expressions into high-quality visual equations within the web interface.

Version control and software development were managed using Git and GitHub, allowing changes to the source code to be tracked, reviewed, and restored throughout the project lifecycle while maintaining a complete history of development. Application data, including user accounts and stored notes, was managed using a lightweight relational database. Development and debugging were performed using Visual Studio Code alongside modern web development tools and browser debugging utilities to support implementation, testing, and refinement of the system.

The combination of these data sources and software tools supported the complete project lifecycle, from requirements gathering and system development to testing and evaluation. The selection of locally deployed AI models was particularly important in satisfying the project's privacy, accessibility, and performance requirements while reducing dependence on external cloud services.

The evaluation methodology was designed to determine whether the INTEGRAL prototype satisfied the functional and non-functional requirements established during the earlier stages of the project. A mixed-methods approach was adopted, combining quantitative performance measurements with qualitative usability assessment to provide a comprehensive evaluation of the system (ISO, 2018; Brooke, 1996; Nielsen, 1994).

Functional evaluation focused on verifying that each implemented feature operated according to its corresponding acceptance criteria defined within the Software Requirements Specification. Core functionality, including user authentication, note management, speech-to-text transcription, AI-assisted LaTeX generation, and file export, was systematically tested using representative user scenarios. The successful completion of each scenario was recorded to assess the overall completeness and reliability of the implemented functionality (Pressman and Maxim, 2020).

The performance of the AI-assisted components was evaluated by measuring response latency, transcription accuracy, and the quality of generated mathematical notation. Particular attention was given to the responsiveness of the locally deployed Mistral 7B and Whisper models, as low processing latency is essential for supporting note-taking during live lectures. Performance testing also considered the computational resources required for local inference to determine whether the selected architecture satisfied the project's real-time requirements (Jiang *et al.*, 2023; Radford *et al.*, 2023).

Usability evaluation was conducted through structured user testing involving representative users from the target audience. Participants were asked to complete predefined note-taking tasks while interacting with the prototype, after which they provided feedback regarding ease of use, interface design, navigation, perceived usefulness, and overall user experience. Where

appropriate, the **System Usability Scale (SUS)** was employed as a standardised instrument for assessing perceived usability and facilitating comparison with established usability benchmarks (Brooke, 1996).

The evaluation also considered non-functional characteristics including accessibility, reliability, maintainability, privacy, and security. Accessibility was assessed against recognised principles of inclusive interface design and compatibility with assistive technologies (ISO, 2019; W3C, 2023). Privacy and security were evaluated by verifying that all AI processing remained local to the user's device and that user authentication and data storage mechanisms operated as intended.

Together, these evaluation activities provided evidence that the developed prototype addressed the project objectives and user requirements while identifying areas for future improvement.

Ethical considerations were integrated throughout the INTEGRAL project, from requirements elicitation and system design to implementation and evaluation. The project was developed in accordance with the ethical principles of respect for participants, privacy, transparency, accessibility, and responsible use of artificial intelligence. These considerations were particularly important because the application was designed for educational use and involved collecting user feedback during the requirements analysis and evaluation stages (British Educational Research Association (BERA), 2024).

Participation in the prospective user questionnaire and system evaluation was voluntary. Participants were informed about the purpose of the project, the intended use of the collected information, and their right to withdraw from participation at any stage without consequence. No personally identifiable information was required beyond that necessary for the analysis, and all responses were anonymised prior to analysis. Data collection and processing were conducted in accordance with the **UK General Data Protection Regulation (UK GDPR)** and the **Data Protection Act 2018**, ensuring that personal information was handled securely and used solely for the purposes of this research (Information Commissioner's Office, 2024).

Privacy was a key design consideration throughout the project. Unlike many AI-assisted educational applications that rely on cloud-based services, INTEGRAL was designed to perform both speech recognition and large language model inference locally. By deploying Whisper and Mistral 7B through Ollama, mathematical notes and spoken input remained on the user's device, significantly reducing the risks associated with transmitting sensitive educational data to third-party services. This architectural decision directly supported the project's non-functional requirements relating to privacy, security, and trustworthiness (European Commission, 2019).

Accessibility was considered from the earliest stages of development to ensure that the application could support a diverse range of learners. Requirements gathered from prospective users informed interface design decisions, while accessibility guidance and inclusive design principles were considered during implementation. Features such as speech input, readable mathematical rendering, responsive layouts, and compatibility with assistive technologies were incorporated to improve usability for users with different learning preferences and accessibility needs (ISO, 2019; Anthology, 2026).

The use of generative artificial intelligence also introduces ethical challenges, including the possibility of inaccurate or misleading outputs. To mitigate these risks, the generated mathematical notation was intended to assist rather than replace users' own understanding. Users retain full control over editing and verifying generated content before incorporating it into their notes. This approach aligns with current recommendations for trustworthy and

human-centred AI, where AI systems augment rather than substitute human decision-making (European Commission, 2019).

Overall, ethical principles informed both the research methodology and the technical implementation of the project, ensuring that INTEGRAL was developed as a secure, accessible, privacy-preserving, and responsible educational application.

2.6 Evaluation Methodology

The evaluation methodology was designed to determine whether the INTEGRAL prototype satisfied the functional and non-functional requirements established during the earlier stages of the project. A mixed-methods approach was adopted, combining quantitative performance measurements with qualitative usability assessment to provide a comprehensive evaluation of the system (ISO, 2018; Brooke, 1996; Nielsen, 1994).

Functional evaluation focused on verifying that each implemented feature operated according to its corresponding acceptance criteria defined within the Software Requirements Specification. Core functionality, including user authentication, note management, speech-to-text transcription, AI-assisted LaTeX generation, and file export, was systematically tested using representative user scenarios. The successful completion of each scenario was recorded to assess the overall completeness and reliability of the implemented functionality (Pressman and Maxim, 2020).

The performance of the AI-assisted components was evaluated by measuring response latency, transcription accuracy, and the quality of generated mathematical notation. Particular attention was given to the responsiveness of the locally deployed Mistral 7B and Whisper models, as low processing latency is essential for supporting note-taking during live lectures. Performance testing also considered the computational resources required for local inference to determine whether the selected architecture satisfied the project's real-time requirements (Jiang *et al.*, 2023; Radford *et al.*, 2023).

Usability evaluation was conducted through structured user testing involving representative users from the target audience. Participants were asked to complete predefined note-taking tasks while interacting with the prototype, after which they provided feedback regarding ease of use, interface design, navigation, perceived usefulness, and overall user experience. Where appropriate, the **System Usability Scale (SUS)** was employed as a standardised instrument for assessing perceived usability and facilitating comparison with established usability benchmarks (Brooke, 1996).

The evaluation also considered non-functional characteristics including accessibility, reliability, maintainability, privacy, and security. Accessibility was assessed against recognised principles of inclusive interface design and compatibility with assistive technologies (ISO, 2019; W3C, 2023). Privacy and security were evaluated by verifying that all AI processing remained local to the user's device and that user authentication and data storage mechanisms operated as intended.

Together, these evaluation activities provided evidence that the developed prototype addressed the project objectives and user requirements while identifying areas for future improvement.

2.7 Ethical Considerations

Ethical considerations were integrated throughout the INTEGRAL project, from requirements elicitation and system design to implementation and evaluation. The project was developed in accordance with the ethical principles of respect for participants, privacy, transparency, accessibility, and responsible use of artificial intelligence. These considerations were particularly important because the application was designed for educational use and involved collecting user feedback during the requirements analysis and evaluation stages (British Educational Research Association (BERA), 2024).

Participation in the prospective user questionnaire and system evaluation was voluntary. Participants were informed about the purpose of the project, the intended use of the collected information, and their right to withdraw from participation at any stage without consequence. No personally identifiable information was required beyond that necessary for the analysis, and all responses were anonymised prior to analysis. Data collection and processing were conducted in accordance with the **UK General Data Protection Regulation (UK GDPR)** and the **Data Protection Act 2018**, ensuring that personal information was handled securely and used solely for the purposes of this research (Information Commissioner's Office, 2024).

Privacy was a key design consideration throughout the project. Unlike many AI-assisted educational applications that rely on cloud-based services, INTEGRAL was designed to perform both speech recognition and large language model inference locally. By deploying Whisper and Mistral 7B through Ollama, mathematical notes and spoken input remained on the user's device, significantly reducing the risks associated with transmitting sensitive educational data to third-party services. This architectural decision directly supported the project's non-functional requirements relating to privacy, security, and trustworthiness (European Commission, 2019).

Accessibility was considered from the earliest stages of development to ensure that the application could support a diverse range of learners. Requirements gathered from prospective users informed interface design decisions, while accessibility guidance and inclusive design principles were considered during implementation. Features such as speech input, readable mathematical rendering, responsive layouts, and compatibility with assistive technologies were incorporated to improve usability for users with different learning preferences and accessibility needs (ISO, 2019; Anthology, 2026).

The use of generative artificial intelligence also introduces ethical challenges, including the possibility of inaccurate or misleading outputs. To mitigate these risks, the generated mathematical notation was intended to assist rather than replace users' own understanding. Users retain full control over editing and verifying generated content before incorporating it into their notes. This approach aligns with current recommendations for trustworthy and human-centred AI, where AI systems augment rather than substitute human decision-making (European Commission, 2019).

Overall, ethical principles informed both the research methodology and the technical implementation of the project, ensuring that INTEGRAL was developed as a secure, accessible, privacy-preserving, and responsible educational application.

3 Implementation

3.1 Architecture overview

INTEGRAL follows a three-tier client-server architecture. A client-server architecture is a way of organising a software system in which one part of the system (the client) requests data or services, while another part (the server) processes those requests and provides the required resources over a network. In a three-tier architecture, the server-side functionality is further divided into three distinct layers, each with a specific responsibility. The presentation layer provides the user interface and handles user interactions, the application logic layer processes requests and implements the system's business rules, and the data persistence layer stores, retrieves, and manages the application's data. This separation of responsibilities makes the system easier to develop, maintain, and extend, as changes to one layer can often be made without affecting the others.

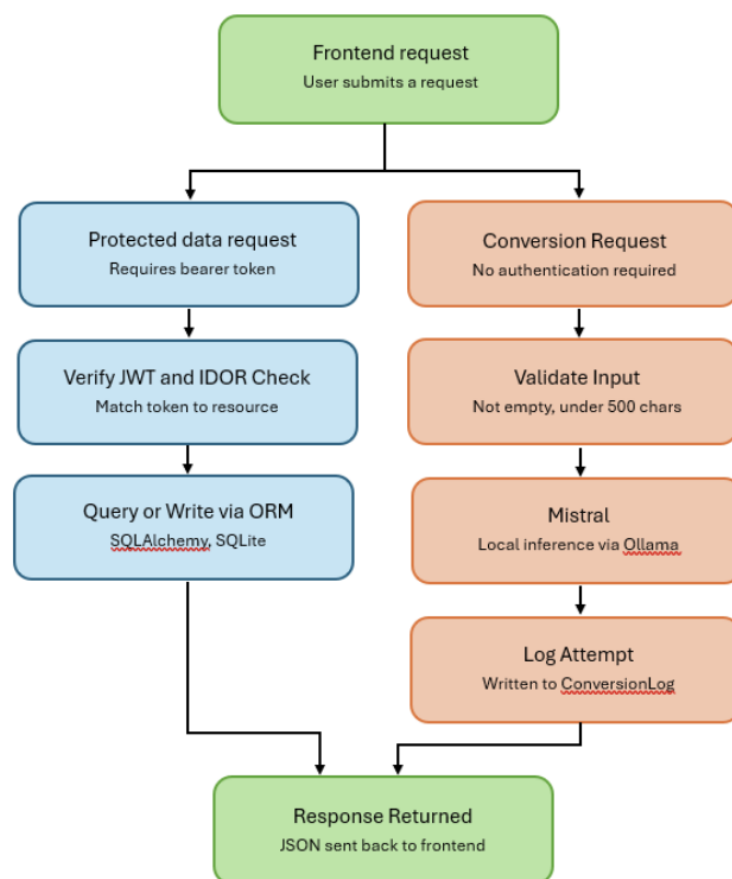


Figure 3.1 System Functionality Diagram

3.1.1 Frontend

The client [see figure 3.1] is a React and TypeScript single-page application, built with Vite and running entirely in the browser with no installation required. It communicates with the backend exclusively through JSON requests over HTTP, which keeps the frontend a thin client with no direct access to the database or the AI models. Every operation is mediated by the backend.

In the current development environment, this communication occurs over HTTP. In production, encryption in transit is expected to be handled at the university server infrastructure level via HTTPS, rather than within the application code itself, a deployment-layer responsibility rather than an application-layer one.

3.1.2 Backend

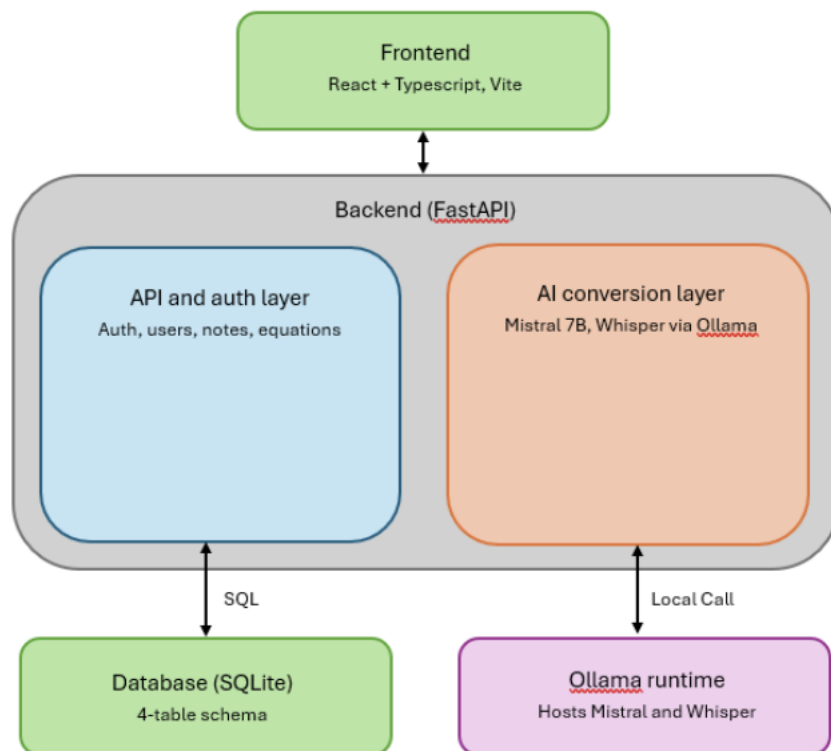


Figure 3.2 System Architecture

The application logic layer is responsible for processing requests, applying the system's rules, and coordinating communication between the user interface and the data storage layer. Within INTEGRAL, this layer is implemented using FastAPI and manages tasks such as routing requests to the appropriate functions, verifying user authentication, and performing data operations such as creating, retrieving, updating, and deleting records (commonly referred to as CRUD operations). Database interactions are handled through SQLAlchemy, an Object-Relational Mapping (ORM) tool that allows application code to work with stored data using programming objects rather than directly executing SQL queries. This abstraction improves

maintainability and provides additional protection against security risks such as SQL injection attacks. The layer also implements a custom authentication system based on JSON Web Tokens (JWTs), which are used to verify user identity, alongside protection against Insecure Direct Object Reference (IDOR) vulnerabilities, ensuring that users can only access resources they are authorised to view.

The AI conversion layer contains the two AI components used by the application: Mistral 7B for text-to-LaTeX conversion, and Whisper for speech-to-text transcription, both served locally through Ollama. Mistral 7B's selection is supported by its published performance characteristics rather than being an arbitrary choice: it has been shown to outperform the previous best 13-billion-parameter model across all tested benchmarks, and to surpass the best 34-billion-parameter model specifically in reasoning, mathematics, and code generation (Jiang et al., 2023). This is directly relevant to INTEGRAL's use case, as LaTeX generation from a mathematical expression is a structured, code-like output task closely tied to mathematical content. Whisper's role is supported by its demonstrated robustness as a speech recognition model. When trained on large-scale, weakly supervised audio data, it generalises well to standard benchmarks in a zero-shot setting, without requiring task-specific fine-tuning, and approaches human-level accuracy and robustness (Radford et al., 2022).

Running both models locally via Ollama, rather than through a cloud-hosted API, is also a defensible architectural decision. Recent work evaluating quantised language models drawn from the Ollama library, deployed on resource-constrained edge hardware, demonstrates that this is an actively studied deployment pattern with well-characterised energy and latency trade-offs, rather than simply a convenience shortcut (Husom et al., 2025). This local-first approach also aligns with INTEGRAL's zero-install, lab-computer-friendly design goal, as it avoids dependence on external API availability or per-request cost, and keeps sensitive user data from being transmitted to a third-party service.

3.1.3 Database

INTEGRAL uses a SQLite database managed through SQLAlchemy, with four tables storing the core application data.

The 'users' table stores one record per registered account, containing a UUID primary key, the user's display name, email address, bcrypt password hash, appearance preference, and a creation timestamp. A record is written to this table when a user successfully registers via 'POST /api/auth/register/'. The password hash is the only sensitive field and is never returned in any API response.

The 'notes' table stores the notes created by each user. Each record contains a UUID primary key, a foreign key linking it to the owning user, a title, the raw plain text content as typed by the user, the processed LaTeX output returned by Mistral (stored in 'noteContentProcessed'), and created and updated timestamps.

The 'conversion_logs' table records every call to the '/api/convert/' endpoint, regardless of whether it succeeded or failed. Each record stores a UUID primary key, the input expression, a boolean success flag, the latency in milliseconds, and a timestamp. This table supports the

performance monitoring requirement, providing empirical data against the p95 latency target of 1.5 seconds.

3.1.4 Component interaction and data flow

A typical request begins at the frontend, which sends a JSON payload over HTTP to a specific FastAPI route. From there, the path a request takes through the backend depends on what it is asking for. Requests that touch personal data, such as retrieving a user's notes or updating account settings, are routed through the API and authentication layer first, where the request is validated and authenticated before the SQLAlchemy ORM issues a parameterised query against SQLite and returns the result as JSON. Requests that ask for a mathematical expression to be converted, by contrast, bypass the database entirely on their way in and are routed instead to the AI conversion layer, with the outcome of that conversion only touching the database afterwards, when the attempt is written to the ConversionLog table. This means the two layers of the backend are used somewhat independently depending on the nature of the request, even though both ultimately return a JSON response to the same frontend client.

3.1.5 Authentication and protected endpoints

The initial plan for INTEGRAL was to use Microsoft Single Sign-On (SSO), which would have allowed students and staff to log in using their existing University of Westminster credentials. This was chosen for its convenience and alignment with university infrastructure. However, during implementation it became clear that university Microsoft accounts do not have permission to register applications on the Azure Portal, which is a required step for enabling Microsoft SSO in any third-party application. Attempts to work around this by registering through a personal Microsoft account introduced further problems around sharing access and managing credentials securely across the team.

Given these constraints and the project timeline, the decision was made to build a custom JWT-based authentication system instead. A new account is created via 'POST /api/auth/register/', which requires a university email address ending in '@westminster.ac.uk', along with a display name and password. The password is hashed with bcrypt before being stored and is never stored or logged in plain text. Logging in via 'POST /api/auth/login/' verifies the submitted credentials and, if successful, issues a JWT token valid for eight hours. This token must be included as a Bearer token in the Authorization header of any subsequent request to a protected endpoint.

The main known limitation of the custom system is that JWT tokens remain valid for up to eight hours after logout, since there is no server-side token revocation. This is documented as recommended future work, alongside a token refresh system to handle session extension without requiring users to log in again.

Every protected endpoint requires this token to be included as an Authorization: Bearer <token> header. On each request to a protected endpoint, the backend decodes the token using a secret key and extracts the user's ID from it. For endpoints that access personal data specifically, such as a user's notes or profile, the backend additionally checks that this extracted user ID matches the ID of the resource being requested. This is the system's protection against insecure direct object reference (IDOR) attacks, and it prevents one

authenticated user from accessing another user's notes simply by changing an ID in the request URL. A request that fails either check, an invalid or missing token, or a mismatch between the token's user and the requested resource, is rejected with a 401 response before it reaches the database.

Not all endpoints require this authentication. Registration and login are, by necessity, unauthenticated, since a token cannot exist before login. The conversion and transcription endpoints are unauthenticated as well, which follows from their design as stateless utility endpoints that do not read or write personal data unless a result is subsequently saved as a note.

This layered separation is significant for two reasons. First, it means the AI conversion layer is fully decoupled from the API and authentication layer, which, as the following section explains, allows both of INTEGRAL's input pipelines to converge on a single, isolated point in the system regardless of their entry method. Second, it allows the AI layer to degrade independently of the rest of the system. The fallback behaviour, in which an unavailable Ollama instance returns a plain text wrapper instead of causing a failure, does not affect authentication, note management, or export functionality elsewhere in the application.

3.2. Input methods

Building on this architecture, INTEGRAL supports two input modes, text and speech, both of which are designed to converge on the AI conversion layer described above, rather than being handled as separate systems. This convergence is a deliberate architectural choice. It means the system requires only one implementation of the mathematical expression-to-LaTeX conversion logic, regardless of how the expression enters the system.

For text input, the user types a plain text expression into the frontend editor, which is sent to the backend via 'POST /api/convert/'. Before the input reaches the model, it passes through validation in the API and authentication layer to confirm that it is not empty, is under 500 characters, and is not a bare number. These checks prevent wasted calls to the language model, which improves the responsiveness of the user experience, reduces unnecessary computational load, and supports the broader sustainability goals of the project. This is consistent with findings in the wider literature on inference efficiency, where the proper application of relevant efficiency optimisations has been shown to reduce total energy use by up to 73% compared with unoptimised baselines (Fernandez et al., 2025).

For speech input, the user's spoken expression is recorded in-browser and sent to 'POST /api/transcribe/'. As noted in the architecture section above, this request is handled by Whisper (base model) within the AI conversion layer, which transcribes the audio to plain text. Whisper relies on ffmpeg for audio processing. Once transcribed, this plain text is passed into the same validation and conversion pipeline used by text-based input, converging with the text pathway at the point where the expression is sent to Mistral for LaTeX conversion. It is worth noting that validation at this stage only prevents a wasted call to the conversion model, not to Whisper itself, since transcription must occur before the resulting text can be validated. Even so, this step still protects output quality, as it catches a poor or empty transcription before it reaches the conversion model, rather than allowing a nonsensical LaTeX output to be generated and shown to the student.

Both pipelines call `'convert_with_mistral()'`, which sends the validated expression to the Mistral 7B model within the AI conversion layer. Mistral returns raw LaTeX, which is then cleaned, markdown backticks and stray dollar signs are removed, and explanatory text is filtered out. The result is cached in memory so that repeated expressions skip a fresh model call, further reducing unnecessary inference. Errors are handled gracefully throughout the process. If Ollama is unavailable, the system returns a plain-text wrapper rather than failing outright, consistent with the graceful-degradation behaviour of the AI conversion layer described above.

Every conversion attempt is logged to the ConversionLog table, recording the input text, a success flag, and latency in milliseconds. This data is used to monitor progress against the project's target of a p95 latency of 1.5 seconds or less for LaTeX conversion. Once conversion is complete, the final LaTeX is returned to the frontend and rendered in the preview panel.

In this way, both input modes, although different in their entry points, are processed through the single shared backend pipeline described above, which is consistently validated, monitored, and cached.

Three input modes were originally planned for INTEGRAL, text, speech, and handwritten expression recognition. However, the handwritten input mode was not implemented within the project timeline due to time constraints. The student team intends to continue development beyond the official project period, with handwriting-to-LaTeX conversion planned as a future addition.

3.3 LaTeX generation process

The LaTeX generation pipeline constitutes the core functional component of the INTEGRAL system. Upon submission of a mathematical expression, whether through typed text input or voice dictation, the input is transmitted to a locally hosted Mistral 7B Instruct model via the Ollama inference framework. The model performs natural language translation to a LaTeX conversion, returning syntactically valid LaTeX notation that is subsequently rendered within the note editor as a formatted mathematical preview.

The model selection of Mistral-7B Instruct as the underlying model was informed by a systematic comparative analysis of locally deployable large language models (LLMs). Figure 2.2.1 benchmark evaluations across multiple dimensions including MMLU, logical reasoning, knowledge, and comprehension demonstrate that Mistral achieves performance equivalent to models 2-5 times its parameter count as illustrated in Figure 5.1.1. This efficiency characteristic renders the model particularly well suited for deployment on the institutional hardware without necessitating dedicated GPU infrastructure or cloud-based API dependencies. Significance to the deployment context is the data privacy consideration, which was mentioned in section 2.1, ensuring the transmitted data never reaches external third-party services such as cloud-based alternatives OpenAI API.

To produce accurate and syntactically consistent LaTeX outputs, it's important to construct a necessitated prompt engineering methodology [Brown et al., 2020]. Initial experimentation with zero shot prompting approach, where the model only receives the raw mathematical expression with no contextual instruction or output format guidance. In zero-shot prompting, no examples of the desired input/output behaviour are provided, thus the model must infer the

expected response format from the instruction alone. In return, this approach yielded inconsistent results: the model frequently generated superfluous explanatory proeses, introduced model dollar sign delimiters (\$...\$), and produced markdown code block constructs (``latex``) alongside the LaTeX output, which requires extensive post-processing to extract usable notation. To address these shortcomings, a few-shot prompting strategy [Brown et al., 2020] was subsequently adopted, where the system prompt was augmented with curated input/output exemplars demonstrating this precise format required. This approach leverages the model's context learning capability, enabling it to infer the expected output style from provided examples rather than relying explicit rules alone.

To assess the quality and consistency of the LaTeX generation, a formal evaluation was conducted utilising three complementary quantitative metrics applied against a standardised test set of 15 mathematical expressions derived primarily from 'Mathematics for Computing' module slides of University of Westminster. The test set was designed to provide representative coverage across mathematical domains, encompassing basic algebra, differential and integral calculus, number theory, and linear algebra.

```

Input:    30 factored into prime numbers
LaTeX:    30 = 2^2 \cdot 3 \cdot 5
Latency:  10720ms

Input:    420 factored into prime sized blocks
LaTeX:    420 = 2^3 \cdot 5 \cdot 7 \cdot 13
Latency:  2714ms

Input:    945 factored into prime numbered chunks
LaTeX:    945 = 3^2 \cdot 5 \cdot 7 \cdot 13
Latency:  2446ms

Input:    fundamental theorem of arithmetic
LaTeX:    n = \prod_{i=1}^k p_i^{a_i}
Latency:  29ms

Input:    prime factorisation of n
LaTeX:    n = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_k^{a_k}
Latency:  2ms

Input:    if p divides ab then p divides a or p divides b
LaTeX:    p \mid ab \Rightarrow p \mid a \text{ or } p \mid b
Latency:  2098ms

Input:    every integer n greater than 1 can be written as product of primes
LaTeX:    n = \prod_{i=1}^k p_i^{a_i}
Latency:  18ms

Average latency: 2575ms

```

Figure 3.3 Mistral-7B Latency Results

The table [see figure 3.3] shows terminal output demonstrating the LaTeX generation pipeline evaluated against seven expressions derived from the University of Westminster Mathematics for computing Module 4 content on the Fundamental Theorem of Arithmetic.

3.4 UI/UX

The user interface was developed as a single-page application using React 19 and TypeScript, with Vite as the build environment and Tailwind CSS for styling. Mathematical rendering was implemented through a react-markdown pipeline

augmented with remark-math and rehype-katex, allowing raw learner input to be typeset in real time using the KaTeX engine. This technical foundation was selected to satisfy a central design requirement: that the interface should present formal mathematical notation with typographic fidelity while remaining responsive enough for use during live lectures and seminars. Consistent with the project's non-functional requirements, interface responsiveness was treated not as a cosmetic concern but as a factor shaping how effectively students adapt digital tools to authentic lecture environments (Arden et al., 2024).

The interface adopts a two-panel layout in which learners enter freeform text into an editor on one side and view the rendered mathematical output on the other, as illustrated in **Figure 3.1**. This spatial separation was intended to preserve a clear correspondence between informal input and formal representation, reflecting the distinction (emphasised throughout this work) between mathematical representation and mathematical reasoning. Rather than obscuring the transformation from natural language to LaTeX, the design deliberately exposes it, positioning the system as a representational mediation layer that supports, rather than replaces, the learner's own construction of meaning (Martin et al., 2025). Supporting components, including a collapsible and searchable note sidebar, an export menu, and a reusable library of common equations, were introduced incrementally to reduce navigational and organisational overhead without crowding the primary editing surface (Choi and Seo, 2024).

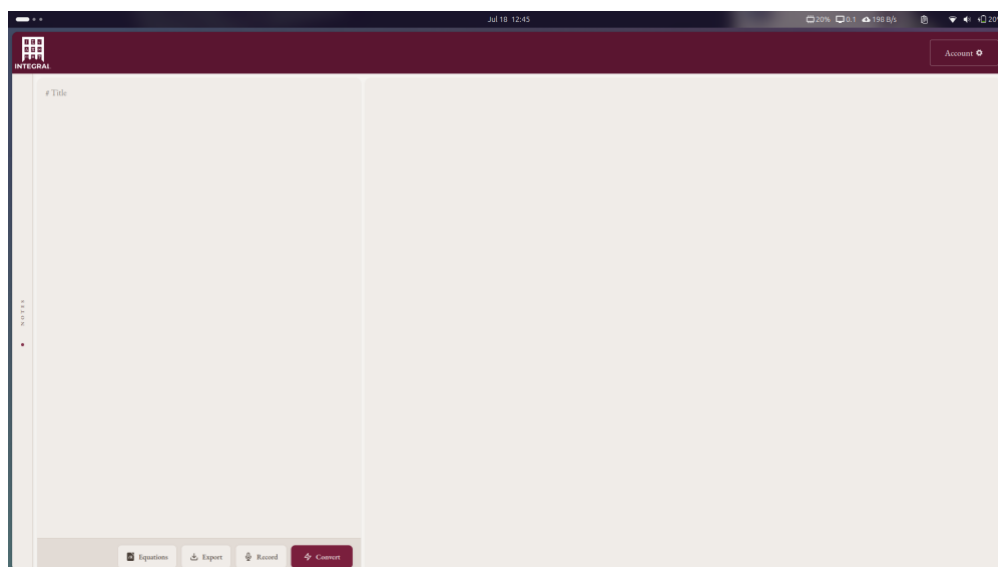


Figure 3.4: The two-panel editor interface in light mode, showing the input area (left) and rendered output panel (right), with the Equations, Export, Record, and Convert action bar at the bottom.

A central pedagogical decision concerned the mechanism by which conversion is triggered. An early prototype performed conversion automatically on a short debounce, re-rendering the output continuously as the learner typed. In practice this approach produced visual flickering, imposed unpredictable latency, and generated frequent model invocations that were both computationally wasteful and cognitively distracting. The design was therefore pivoted so that conversion is initiated explicitly by the learner through a dedicated Convert button, shown prominently in **Figure 3.2**. Beyond its technical benefits in stabilising rendering and reducing redundant inference, this change carries pedagogical significance: it returns control of the

transformation to the learner, preserving a deliberate moment of authorship rather than allowing the system to pre-empt the student's engagement with notation. This aligns the interface with the concern that AI assistance should reduce extraneous load without encouraging the cognitive offloading and dependency associated with diminished higher-order thinking (Martin et al., 2025; Zhang and Jiang, 2026).

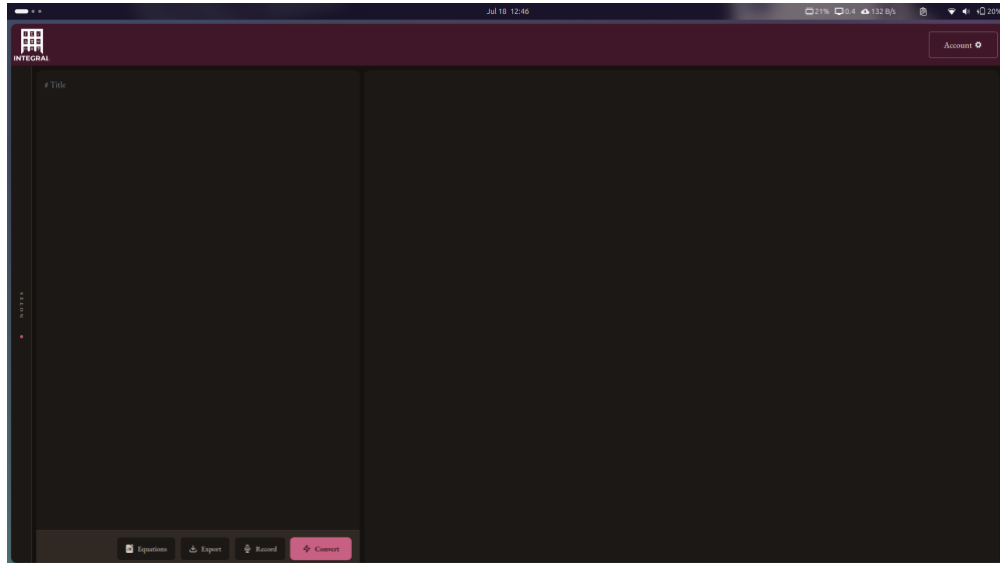


Figure 3.5: The editor interface in dark mode. The prominent Convert button (bottom right) reflects the deliberate decision to make conversion an explicit learner-initiated action rather than an automatic process.

The range of supported input methods was expanded iteratively in response to accessibility considerations. Text entry was complemented by a voice-input facility, implemented through a browser-based audio recording hook connected to a locally deployed speech-to-text service, enabling learners to dictate mathematical content that is then transcribed into the editor. This broadened the modes through which mathematical ideas can be expressed, consistent with universal design principles that seek to reduce unnecessary technical barriers for all learners rather than accommodating a single input paradigm (CAST, 2024; Choi and Seo, 2024). Transcribing mathematical language into structured notation remains a demanding and only recently addressed problem, and the design therefore treated voice input as an assistive complement to, rather than a replacement for, direct text entry (Korzh et al., 2025). Interface states such as recording and transcription were made visually explicit through iconography and animation, ensuring that the system's status always remained legible to the user.

Accessibility and personalisation were further addressed through a configurable theme system. The selected theme is applied immediately in the interface and persisted both locally and to the backend on a per-user basis, so that the preference follows the learner across devices and sessions. This decision reflects the broader interpretation of accessibility adopted in this project, in which equitable access to comfortable and legible working environments is treated as integral to usability rather than as an optional enhancement (Choi and Seo, 2024). The reusable equations library serves a complementary function, allowing learners to insert commonly used expressions without recalling the corresponding LaTeX syntax, thereby directly mitigating the representational friction identified as a key barrier to digital mathematical notetaking (Gkritzali et al., 2024). Such friction is compounded by the fact that mathematical

notation remains difficult for assistive technologies to interpret reliably, underscoring the importance of structured, machine-readable output rather than notation encoded purely as images (Kortemeyer, 2023).

Note management was designed to minimise administrative burden and support fluent capture during learning. As shown in **Figure 3.3**, the collapsible Notes sidebar lists saved notes alongside a search facility, allowing learners to navigate their material without leaving the editing surface. Note titles are derived automatically from the first line of content as the learner types, while an explicit rename action allows this behaviour to be overridden when a custom title is to be preserved. Content persisted to the backend through a debounced auto-save mechanism, and client-side data fetching and cache invalidation were handled through a query library to keep the interface consistent with server state without requiring manual refreshing. Collectively, these behaviours were intended to make the act of notetaking feel continuous and low-friction, in keeping with evidence that students value digital note-taking environments that integrate smoothly into the practical realities of lecture settings (Arden et al., 2024).

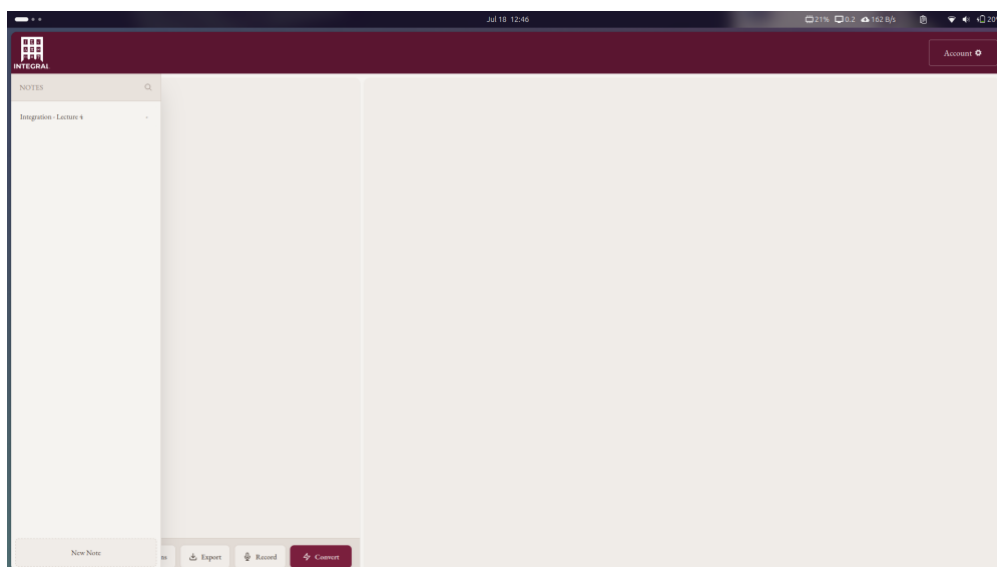


Figure 3.6 The collapsible Notes sidebar in light mode, displaying a saved note ("Integration – Lecture 4") alongside the search facility and the New Note control at the bottom.

Finally, the authentication and entry experience underwent a significant pivot over the course of development. An initial placeholder login was first replaced by a fully implemented authentication flow, incorporating reusable form components, client-side validation constraining registration to institutional email addresses, and protected routing to restrict access to authenticated users. This was subsequently extended towards institutional single sign-on through Microsoft authentication, aligning the entry experience with the university's existing identity infrastructure. The resulting Account settings page, illustrated in **Figure 3.4**, exposes profile management, the light/dark mode preference toggle, and a bulk data export facility, consolidating user-facing configuration in a single location. This progression reflects the project's wider commitment to responsible deployment and data governance: by integrating with institutional identity provision, the interface reduces the friction of separate credential management for students while reinforcing the system's alignment with established data protection expectations (Choi and Seo, 2024).

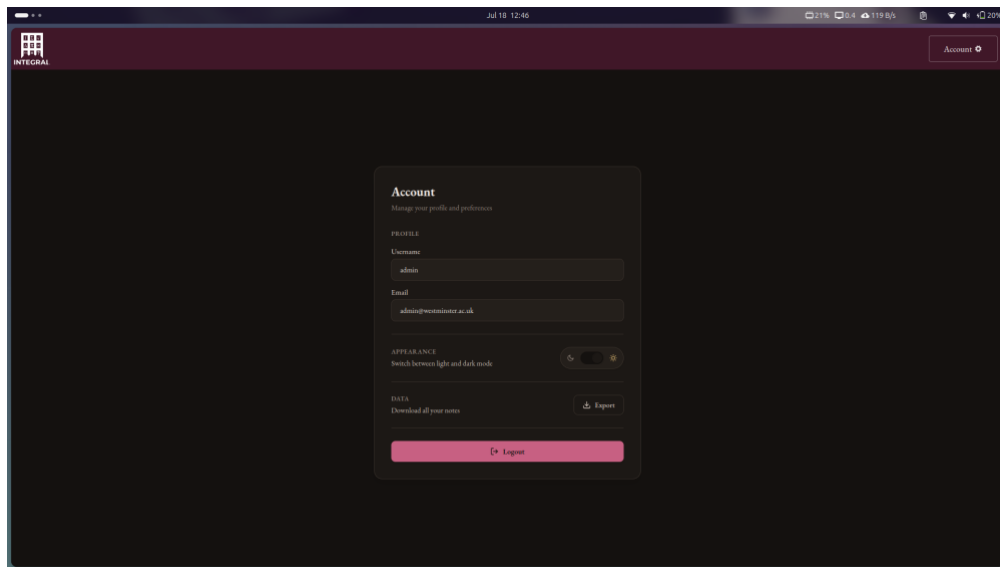


Figure 3.7 The Account settings page, showing the user profile panel, the light/dark mode toggle, and the option to export all notes, reflecting the per-user personalisation and data portability features described in this section.

4 Testing

Testing and evaluation were conducted throughout the development of the INTEGRAL system to verify that the implemented solution satisfied the functional and non-functional requirements defined during the requirements analysis. A comprehensive testing strategy was adopted to assess the correctness, reliability, performance, usability, and accessibility of the application, while also evaluating the effectiveness of the integrated artificial intelligence components. The objective of the testing process was not only to identify software defects but also to determine whether the system met its intended educational purpose of supporting efficient and accessible mathematical note-taking.

A combination of testing methods was employed to provide a comprehensive assessment of the prototype. Functional testing verified the correct operation of individual features and overall system behaviour against the defined acceptance criteria. Performance testing measured the responsiveness and efficiency of the application, with particular emphasis on the latency of the locally deployed AI models. User Acceptance Testing (UAT) evaluated the application from the perspective of prospective users, ensuring that the implemented functionality aligned with user expectations and project objectives. In addition, specialised evaluation was undertaken for the artificial intelligence components, including speech recognition and LaTeX generation, using established quantitative metrics to assess transcription accuracy and the quality of generated mathematical notation.

The testing process was performed iteratively throughout development, allowing issues to be identified and resolved before subsequent development stages. This continuous approach supported incremental refinement of the prototype and reduced the risk of defects propagating into later versions of the system. The findings presented in this chapter demonstrate the extent to which the completed prototype fulfils the project's technical, educational, and usability requirements while highlighting areas for future improvement.

The remainder of this chapter presents the testing methodology, the individual testing activities undertaken, and the evaluation results for each major component of the INTEGRAL system.

Test ID	Test Type	Feature / Area	Requirement ID	Priority	Owner	Status	Linked Sheet	Notes
FT-001	Functional	Login	FR-LOGIN-001	Must	Matt	In Progress	Functional Testing	Valid and invalid login should be covered.
VT-001	Validation/Input	Registration Form	FR-REG-001	Must	Matt	Completed	Validation Input Testing	Check empty and invalid fields.
ST-001	Security	Protected Pages	NFR-SEC-001	Must	Matt	In Progress	Security Testing	Unauthenticated users must be denied access.
UI-001	UI	Navigation	NFR-UI-001	Should		Not Started	UI Testing	Check menu, buttons and feedback messages.
CT-001	Compatibility	Responsive Layout	NFR-COMP-001	Should		Not Started	Compatibility Testing	Check desktop, tablet and mobile views.
PT-001	Performance	Login Speed	NFR-PERF-001	Should	Matt	In Progress	Performance Testing	Record observed response time.
IT-001	Integration	Frontend and Backend	FR-API-001	Must		Not Started	Integration Testing	Check API response and UI update.
UAT-001	UAT	End-to-End User Scenario	UAT-001	Must	Matt	In Progress	User Acceptance Testing	Stakeholder/student acceptance scenario.

Figure 4.1 Master Test Register Table

Testing was organised into a Master Test Register [see Figure 4.1] spanning five categories: functional, validation input, security, performance, and user acceptance.

4.1 Functional Testing (FT)

Test ID	Requirement ID	Feature / Area	Scenario / Test Description	Prerequisites	Test Steps	Test Data / Input	Expected Result	Actual Result	Pass/Fail	Tester	Date	Comments / Evidence
FT-001	FR-LOGIN-001	Login	User logs in with valid credentials	Registered user exists	1. Open login page 2. Enter valid email/password 3. Click login	Valid University test account	User is logged in and redirected to dashboard	User is logged in and redirected to dashboard	Pass	Matt	14/07/2026	
FT-002	FR-REG-001	Registration	New user registration form creates account	User does not already exist	1. Open registration 2. Enter required details 3. Submit form	Name, email, password	Account is created or confirmation is displayed	Account is created and confirmation is displayed	Pass	Matt	14/07/2026	
FT-003	FR-NOTES-001	CRUD	User creates, edits and deletes a note	User logged in	1. Create note 2. Edit note 3. Delete note	Sample text-to-LaTeX note	Create, update and delete actions work correctly	Create, update, and delete actions work correctly	Pass	Matt	14/07/2026	
FT-004	FR-SEARCH-001	Search	User searches saved notes	Saved notes exist	1. Enter keyword 2. Submit search	Keyword from existing note	Relevant notes are displayed	Relevant Notes are displayed	Pass	Matt	16/07/2026	
FT-005	FR-FORM-001	Forms	User submits mathematical input form	User logged in	1. Enter math expression 2. Submit conversion	x squared plus 2x plus 1	LaTeX output is generated and displayed	LaTeX output is generated and displayed	Pass	Matt	14/07/2026	
FT-006	FR-PROFILE-001	Profile	User updates profile details	User logged in	1. Open profile 2. Edit details 3. Save	Display name / preferences	Profile changes are saved correctly	Profile changes are saved correctly	Pass	Matt	14/07/2026	
FT-007	FR-BOOK-001	Booking/Order	Booking/order feature if used by the prototype	Relevant feature enabled	1. Select item/version 2. Confirm booking/order	Sample booking/order data	Booking/order confirmation is shown and stored		N/A	-		

(Figure 4.2 Functional Testing)

In functional testing (FT), core user flows were tested such as registration, login, full note CRUD (Create, Read, Update, Delete), and end-to-end LaTeX conversion, all passing [see Figure 4.2].

FT-005 was particularly important as this is the core function of the application. This tested the application's ability to take user input and use the correct data flow, outlined in the implementation section, to convert the input into correctly formatted LaTeX. This test was performed from the user's point of view, which means we simply typed a number of random plain text mathematical expressions, e.g. "x squared plus 2x plus 1", pressed the 'convert' button, and verified that correctly formatted and correctly rendered LaTeX had been produced by the application. This also confirmed that the output is correctly wrapped for LaTeX rendering. As well as confirming that the Mistral 7B model chosen for this task performs the task it was selected for.

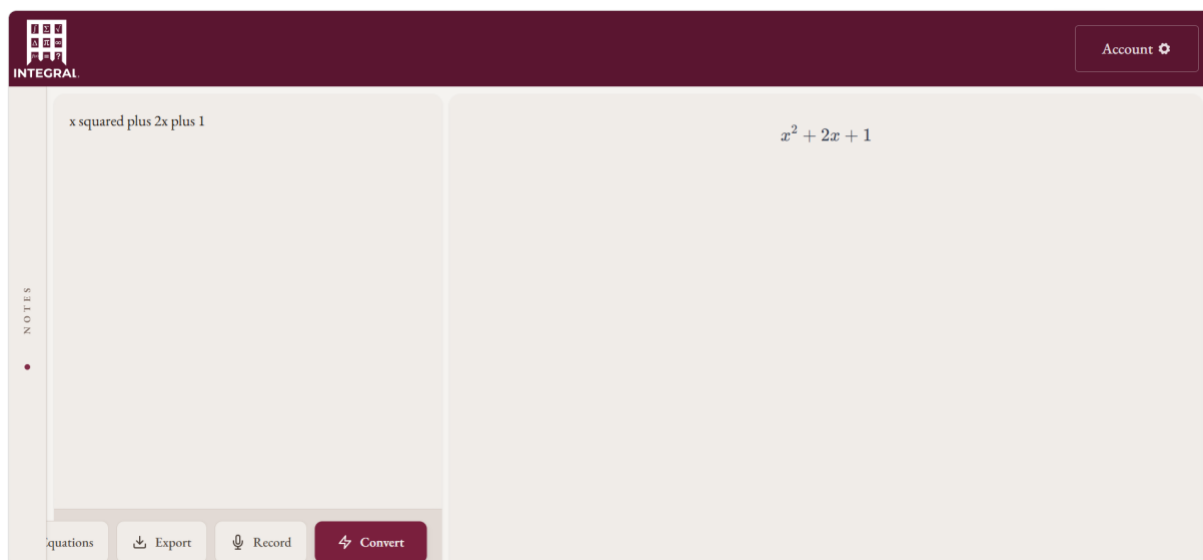


Figure 4.3 Text-to-LaTeX rendering

4.2 Validation Input Testing (VT)

Validation & Input Testing											
Tests how the system handles valid and invalid input, including empty fields, invalid email, password rules, character limits, and incorrect formats.											
Test ID	Requirement ID	Input Field / Form	Validation Scenario	Input Value	Expected Validation Result	Actual Result	Pass/Fail	Severity if Failed	Tester	Date	
T-001	FR-REG-001	Email	Invalid email format	student email	Error message displayed; form not submitted	Error message displayed and form not submitted	Pass	✓	✓	Matt	14/07/2026
T-002	FR-REG-001	Required Fields	Empty required fields	Blank name/email/password	Required field errors displayed	Required field errors displayed	Pass	✓	✓	Matt	14/07/2026
T-003	FR-REG-002	Password	Password too short	abc	Password rule error displayed	Password rule error displayed	Pass	✓	✓	Matt	14/07/2026
T-004	FR-NOTES-001	Note Title	Character limit exceeded	Title over allowed limit	Character limit error or truncation handled safely	Note titles over 255 characters are truncated silently	Pass	✓	✓	Matt	14/07/2026
T-005	FR-MATH-001	Math Input	Incorrect/unsupported format	%invalid expression	Clear error message; no crash	No clear error message produced	Pass	✓	✓	Matt	16/07/2026
T-006	FR-MATH-002	Math Input	Empty expression submitted		400 Error: "Expression cannot be empty"	400 Bad Request Expression cannot be empty	Pass	✓	✓	Mantas	15/07/2026
T-007	FR-MATH-003	Math Input	Expression exceeds character limit	501 character string	400 error: "Expression too long"	400 Bad Request Expression too long — maximum 500 characters	Pass	✓	✓	Mantas	15/07/2026
T-008	FR-MATH-004	Math Input	Pure number submitted	12345	400 error: "Please enter a mathematical expression"	400 Bad Request Please enter a mathematical expression, not just a number	Pass	✓	✓	Mantas	15/07/2026
T-009	FR-MATH-005	Math Input	Valid simple expression	x squared plus 2x plus 1	Returns $x^2 + 2x + 1$	200 OK x squared plus 2x plus 1. latex: $x^2 + 2x + 1$	Pass	✓	✓	Mantas	15/07/2026
T-010	FR-MATH-006	Math Input	Valid complex expression	quadratic formula	Returns valid LaTeX	200 OK the quadratic formula: latex: $x_{[1,2]} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	Pass	✓	✓	Mantas	15/07/2026
T-011	FR-MATH-007	Math Input	Unicode mathematical symbols submitted as input	$\int x^2$	Valid LaTeX returned without crash	200 OK — returned $\sum_{i=1}^n x_i^2$ successfully	Pass	✓	✓	Mantas	15/07/2026
T-012	FR-MATH-008	Math Input	Long multi-part mathematical expression submitted	the integral of x squared plus 2x plus 1 plus the derivative of x cubed plus the sum of i from 1 to n plus the quadratic formula	Valid LaTeX returned for all parts of expression	200 OK — returned full align* block with integral, derivative, sum and quadratic formula all correctly converted	Pass	✓	✓	Mantas	15/07/2026

Figure 4.4 Validation Input Testing Table

In validation input testing (VT), we mainly tested user input validation in fields such as registration. Where a user cannot create an account without using a '@westminster.ac.uk' email address and enforcing an 8+ character password policy. As well as enforcing a character limit on note titles, which will silently truncate any note title over 255 characters [see Figure 4.4].

VT-005 caused us a problem [see Figure 4.5]. This test consisted of submitting a deliberately malformed input ('%%invalid expression') to check whether the system rejects nonsensical mathematical input before conversion. The system failed this test as it did not return a clean error message or reject the input. We found that the backend's existing validation only checks three things: the input is not empty, the input is under 500 characters, and the input is not a bare number. None of these catch structural or semantic invalidity. The malformed string passed all three checks and was forwarded to Mistral, which returned a non-useful LaTeX attempt rather than a clear rejection.

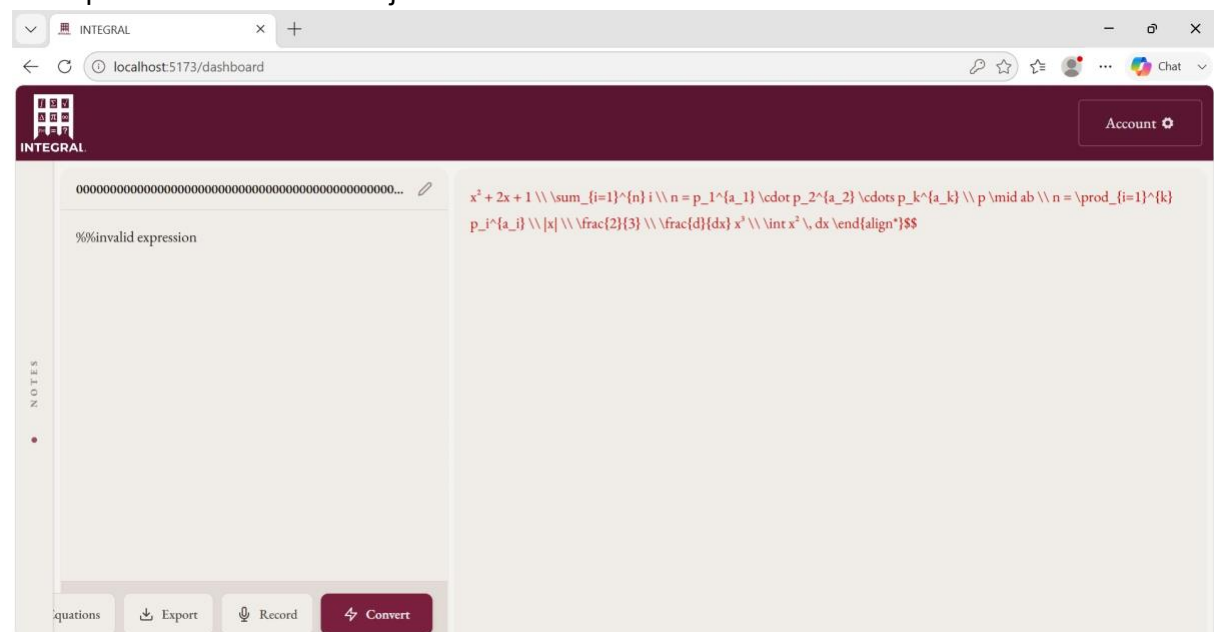


Figure 4.5 Failed Test Run VT-005

A fix was implemented to disallow input strings containing invalid operators such as '%%%' in the test case [see Figure 4.6]:

```
# check for invalid characters – reject input containing non-math symbols
invalid_chars = ['%%%', '###', '@@@', '!!!']
if any(char in expression for char in invalid_chars):
    raise ValueError("Invalid expression – unsupported characters detected")
```

Figure 4.6 VT-005 Code Snippet Fix

This fix works against the test case input and now gives the user a clear error message rather than returning non-useful output, wasting model calls and the user’s time [see Figure 4.7].

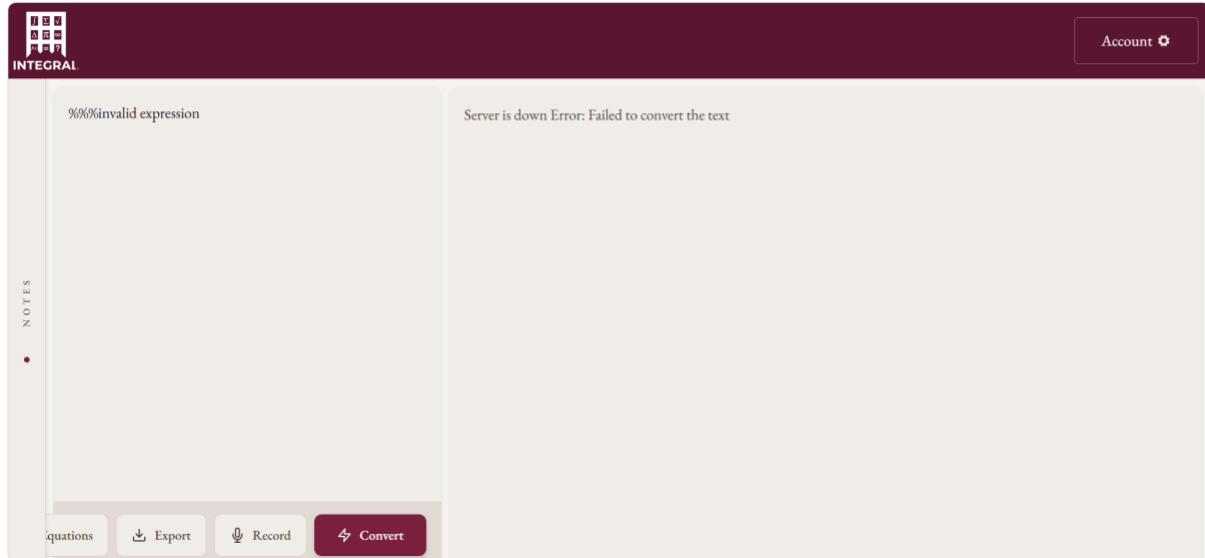


Figure 4.7 Fixed VT-005 Test Case

Distinguishing malformed input from valid-but-informally-phrased mathematical language is a huge natural language processing problem (Jurafsky and Martin, 2025). This is a relatively naive symbol-based filter, which risks false-rejecting legitimate input in a few certain use cases.

The future solution would likely involve a lightweight pre-parser to catch structurally invalid input before it reaches Mistral, or tailoring Mistral’s own response to detect and flag low-confidence outputs after the fact. This is open for discussion by the team in the future.

Although this is not a catastrophic failure, this remains very important to the team as it is a small, missed efficiency opportunity. Since malformed requests still consume a full model call for output that was never going to be useful.

4.3 Security Testing (ST)

Security Testing											
Tests authentication, authorisation, protected pages, SQL injection, XSS, password handling, and session management.											
Test ID	Requirement ID	Security Area	Test Description	Attack / Test Input	Expected Result	Actual Result	Pass/Fail	Risk Level	Tester	Date	Comments / Evidence
ST-001	NFR-SEC-001	Authentication	Access protected page without login	Direct URL to dashboard	Access denied; redirected to login	Access denied and redirected to login when trying to access /dashboard	Pass	Low	Matt	14/07/2026	Redirected to login page
ST-002	NFR-SEC-002	Authorisation	User attempts to access another user's note	Change note ID in URL/API request	Access denied; no data leakage	Access denied; no data leakage	Pass	Low	Matt	14/07/2026	Access denied
ST-003	NFR-SEC-003	SQL Injection	Attempt SQL injection in login/search field	' OR '1'='1	Input rejected/sanitised; no unauthorised access	Input rejected/sanitised; no unauthorised access	Pass	Low	Matt	14/07/2026	SQL string is stored as plaintext in notes and exports, not executed
ST-004	NFR-SEC-004	XSS	Attempt script injection in note/form	<script>alert('x')</script>	Script is escaped/sanitised and not executed	Script is escaped/sanitised and not executed	Pass	Low	Matt	14/07/2026	Script is stored and exported as plain text in notes and exports, not executed
ST-005	NFR-SEC-005	Session Handling	Session expires or logout is used	Logout then press back / revisit protected URL	Session invalidated; protected page not shown	Token remains valid for up to 8 hours after logout. Logout clears the token client-side only	Fail	Medium	Matt	14/07/2026	A captured token was successfully reused against a protected endpoint after logout, returning 200. JWT expiry reduced from 24 hours to 8 hours as a partial mitigation. Full server-side token revocation documented as future work.
ST-006	NFR-SEC-006	Password Handling	Check password storage behaviour	Inspect DB/logs where allowed	Passwords are not stored or logged in plain text	Passwords are not stored or logged in plaintext	Pass	Low	Matt	14/07/2026	bcrypt hashes are used for storing passwords in the database

Figure 4.8 Security Testing Table

In security testing (ST), we mainly tested authentication, authorisation, protected pages, SQL injection, XSS, password handling, and session management. Most of these tests passed cleanly, protected endpoints correctly return 401 without a valid token, users cannot access other users' notes, SQL injection attempts are blocked by the SQLAlchemy ORM, and passwords are confirmed to be hashed with bcrypt in the database [see Figure 4.8]. Two tests are worth discussing in detail, as both revealed real issues that had to be addressed.

Created: 2026-07-14 20:22:00 Last Updated: 2026-07-14 20:22:00 This page says xss Raw Input Processed LaTeX	Created: 2026-07-14 20:22:00 Last Updated: 2026-07-14 20:22:00 <script>alert('xss')</script> Raw Input Processed LaTeX
---	--

Figure 4.9 XSS ST-004 Test Case

ST-004 tested for XSS by submitting a script tag ('<script>alert('xss')</script>') into a note's title and content fields. The system failed this test on the first attempt. While the script tag was stored and displayed safely as plain text within the application itself, we found that it did not escape when included in a note's HTML export, meaning the script would execute if the exported file was opened in a browser [see Figure 4.9]. This was a genuine vulnerability rather than a theoretical one, since HTML export is a real, user-facing feature of the application. We fixed this by applying `html.escape()` to user-submitted content before it is inserted into the HTML export template, ensuring any HTML or script content is rendered as literal text rather than executable markup. We retested afterwards using the same payload and confirmed the script tag now appears safely as visible text in the exported file rather than executing. We also confirmed that this vulnerability was specific to the HTML export format, since LaTeX, DOCX, and PDF exports do not execute HTML and were never at risk.

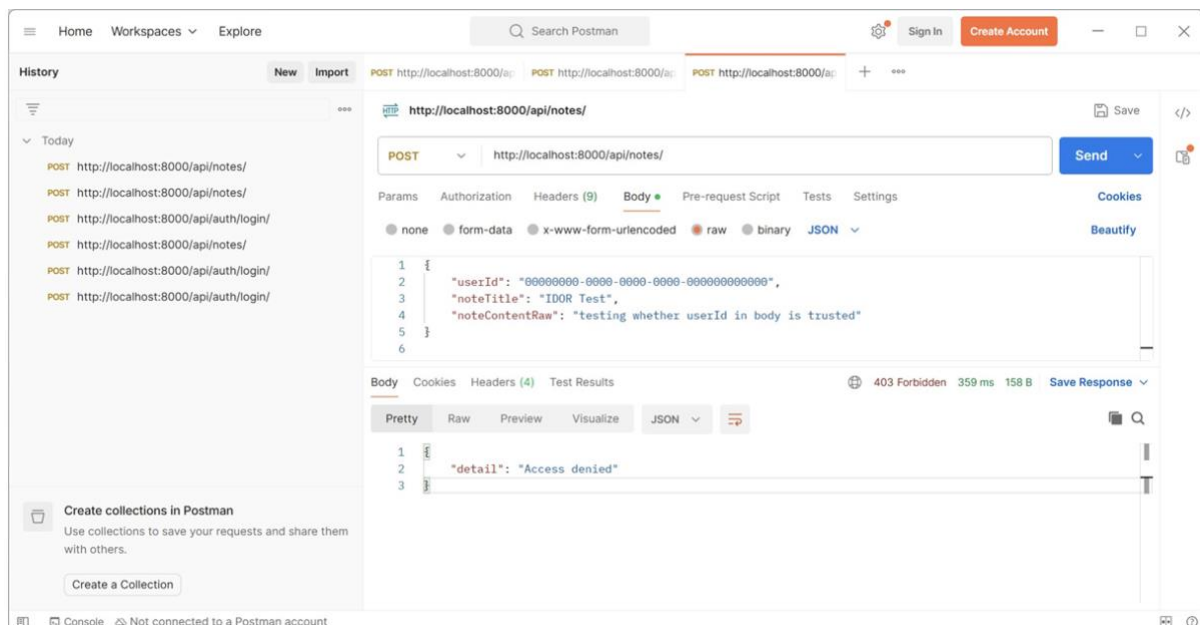


Figure 4.10 ST-005 Test Case

ST-005 tested session handling by logging out via the UI and then reusing the same JWT token, captured beforehand, directly against a protected endpoint. The system failed this test, as the token remained valid and successfully created a note despite the user having already logged out. We found that this is a result of the JWT authentication system being stateless by design, logging out only clears the token on the client side, and there is no server-side mechanism to revoke a token before it naturally expires. Unlike ST-004, this is not something that can be fully resolved with a small code change, since it is a structural property of the authentication approach chosen for the project. Given time constraints, we implemented a partial mitigation rather than a full fix: reducing the token's expiry window from 24 hours to 8 hours, chosen specifically to match the length of a typical student's day on campus. This does not stop a token from being reused after logout, but it meaningfully shrinks the window in which this is possible. We consider this a Medium risk rather than a High risk, since exploiting it requires possession of an already-obtained token rather than allowing broader unauthorised access, though the risk is not negligible given that INTEGRAL is intended for use on shared university lab computers, where a still-valid token on an unattended session is a realistic concern. A full resolution would require a token refresh and revocation system, which we have documented as future work, and which would allow shorter-lived tokens without forcing students to log in repeatedly throughout the day.

4.4 Performance Testing (PT)

Performance Testing										
Tests page loading, database response, search speed, login speed, and behaviour with multiple users.										
Test ID	Requirement ID	Performance Area	Scenario	Target / Threshold	Measured Result	Pass/Fail	Test Environment	Number of Users / Runs	Tester	Date
PT-001	NFR-PERF-001	Page Loading	Dashboard page load time	<= 3 seconds	Average 475ms	Pass	Local/dev server	5 runs	Matt	14/07/2026
PT-002	NFR-PERF-002	Database Response	Retrieve saved notes	<= 2 seconds	Average 95ms	Pass	Local/dev database	5 runs	Matt	14/07/2026
PT-003	NFR-PERF-003	Search Speed	Search notes by keyword	<= 2 seconds	Instantaneous search result in the UI	Pass	Local/dev server	5 runs	Matt	16/07/2026
PT-004	NFR-PERF-004	Login Speed	Login response time	<= 2 seconds	Average 402ms	Pass	Local/dev server	5 runs	Matt	14/07/2026
PT-005	NFR-PERF-005	Multiple Users	Simulated concurrent use	No failure under agreed test load	11 requests completed, 0 failures, 0% error rate: avg: 26.8s, min: 17.97s, max: 36.87s, p95: 35.59s	Pass	Test environment	e.g., 5-10 users	Mantas and Matthew	16/07/2026
PT-006	NFR-PERF-006	LaTeX Conversion	Single expression cold start	< 30 seconds	18482ms (18.4 seconds) cold start	Pass	Local/dev server	3 runs	Mantas	15/07/2026
PT-007	NFR-PERF-007	LaTeX Conversion	Cached expression response time	< 30ms	Average 2ms across 3 runs (Run 1: 2ms, Run 2: 2ms, Run 3: 2ms)	Pass	Local/dev server	3 runs	Mantas	15/07/2026
PT-008	NFR-PERF-008	Whisper transaction	Audio file processing time	< 10 seconds	Average 2173ms across 3 runs (Run 1: 3683ms, Run 2: 1616ms, Run 3: 1222ms)	Pass	Local/dev server	3 runs	Mantas	15/07/2026

Figure 4.11 Performance Testing Table

In performance testing (PT), we tested page loading, database response, search speed, login speed, LaTeX conversion timing, and behaviour under multiple concurrent users. All eight tests in this category passed [see Figure 4.11]. Page loading averaged 475ms against a 3 second target, database retrieval of saved notes averaged 95ms against a 2 second target, search returned results instantaneously in the UI against a 2 second target, and login response time averaged 402ms against a 2 second target. We also tested LaTeX conversion timing specifically: a cold-start conversion (the first request after the model has not recently been used) took 18.4 seconds against a 30 second target, while a cached expression returned in an average of 2ms across three runs, several orders of magnitude faster, confirming the in-memory caching layer discussed in our Input Methods section is working as intended. Whisper transcription of an audio file averaged 2173ms against a 10 second target.

PT-005 is worth discussing in more detail, as it is the test that most clearly reveals the practical trade-offs of our chosen architecture. This test simulated multiple concurrent users using the load testing tool k6, sending simultaneous requests to the conversion endpoint over a sustained window, with a target of no failure under the agreed test load. The system passed this test cleanly: 11 requests completed with 0 failures and a 0% error rate, and the system remained fully stable throughout, with response times averaging 26.8 seconds (min 17.97s, max 36.87s, p95 35.59s). While these response times are far higher than what a single, uncontended request would take, no requests failed or crashed under concurrent load, meeting PT-005's actual stated criterion. We found that this latency is a direct consequence of local hardware constraints rather than a defect in the application: the test machine had only 8GB of RAM shared between the operating system, the 4.4GB Mistral 7B model, and 8 concurrent virtual users all competing for the same limited memory, forcing requests to queue and swap far more heavily than they would on properly provisioned hardware.

Rather than treating this as a defect requiring urgent architectural change, we consider the severity here proportionate to the current prototype stage: no requests failed, and the system degraded gracefully under load rather than crashing. We expect these results to improve significantly on dedicated server infrastructure with adequate memory headroom, and this is documented as recommended future work ahead of any production deployment. It is also worth noting that our decision to run inference locally via Ollama, rather than through a third-party cloud AI API, was made deliberately with GDPR compliance in mind, since routing student data through an external, potentially non-UK/EU-based provider raises data protection concerns that local deployment avoids entirely. We consider this latency trade-off an acceptable and justified cost of that decision at the current stage, rather than a reason to reconsider the underlying architecture.

4.5 User Acceptance Testing (UAT)

Figure 4.12 User Acceptance Testing Table

User Acceptance Testing											
Checks whether the website meets user and stakeholder requirements through realistic scenarios.											
UAT ID	User / Stakeholder Type	Requirement / Need	Scenario	Task Given to User	Acceptance Criteria	Observation Notes	User Feedback	Pass/Fail	Improvement Recommendation	Facilitator	Date
UAT-001	Student tester	Create readable mathematical notes	Student uses the app to convert a simple expression	Convert a typed mathematical expression to LaTeX and save it	User completes task without help; output is readable	User typed '4 over 8 plus 2' and the app produced properly formatted LaTeX, rendered in the preview panel		Pass		Matt	15/07/2026
UAT-002	Student tester	Use speech input if enabled	Student records a spoken equation	Generate LaTeX from spoken input	Output is understandable and errors are manageable	User spoke "four over eight plus two", the app produced properly formatted LaTeX, rendered in the preview panel		Pass		Matt	15/07/2026
UAT-003	Academic/stakeholder	Assess teaching usefulness	Academic reviews generated notes	Review output for clarity and feedback usefulness	Academic confirms whether output supports teaching/feedback needs						

User acceptance testing (UAT) was conducted to evaluate whether INTEGRAL met user and stakeholder needs through realistic usage scenarios, rather than isolated technical tests [see Figure 4.12]. Two core workflows were evaluated. UAT-001 required a student tester to convert a typed mathematical expression into LaTeX and save the result, with success criteria based on completing the task without assistance and producing readable output. This test passed: the tester entered “4 over 8 plus 2”, and the system generated correctly formatted LaTeX that was rendered successfully in the preview panel. UAT-002 evaluated spoken input conversion, requiring a student tester to generate LaTeX from a spoken mathematical expression. This also passed, with the spoken input “four over eight plus two” being converted into correctly formatted and rendered mathematical notation.

Beyond these pass/fail criteria, UAT provided broader feedback on usability, interface design, and potential improvements. One significant issue identified was the handling of multiple expressions within a single note. The tester found that entering separate expressions separated by blank lines caused them to be combined into a single conversion rather than being treated as independent inputs. This occurs because the current conversion pipeline sends the entire note as one block of text to Mistral, without first identifying boundaries between separate expressions or sections of content. As a result, unrelated text or multiple mathematical expressions may be interpreted as one combined instruction, producing an incorrect or incoherent output. Since this limitation was identified independently by a user during testing, it represents a genuine usability issue rather than an isolated technical edge case. A more robust solution would require input segmentation before conversion, allowing separate blocks of content to be processed independently, and this has been documented as future work.

The remaining feedback primarily concerned interface usability and refinement. Several frontend issues were identified, including buttons being partially inaccessible due to insufficient responsiveness, the inability to exit the Account screen once opened, and unclear note editing behaviour where selecting a note title does not allow renaming without using a separate edit function. The tester also suggested improvements to placeholder behaviour, including replacing the “#Title” placeholder with clearer wording and automatically clearing or highlighting placeholder text when editing begins. Additionally, the export buttons were identified as incomplete functionality requiring further development.

The tester also suggested larger interface improvements, including a note management hub similar to the approach used by Google Docs, where users can view, create, and access notes after logging in rather than being taken directly into an editing interface. The current behaviour was considered unclear, as the initial input area does not create a saved note until further actions are taken. More generally, the tester commented that the interface currently feels visually unfinished, describing it as “like looking at a prototype instead of a finished product”. These observations are consistent with the current development stage and provide useful direction for future refinement.

Separately, testing an edge case involving tetration highlighted a limitation in the current mathematical output representation, which does not scale effectively to very large numbers and requires further investigation [see Figure 4.13]. Despite these limitations, feedback on the

core functionality was positive, with the tester describing the system as “usable” and stating, “I like this, it’s solid.”

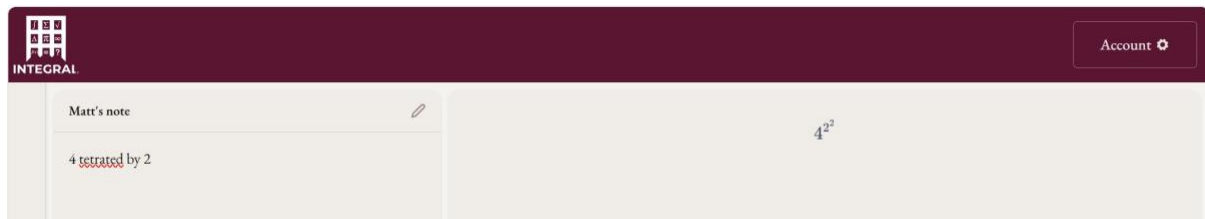


Figure 4.13 Titration Testing Cases

5 Evaluation

The accuracy of the LaTeX generation pipeline was accessed using three complementary quantitative metrics applied against a standardised test set of 15 mathematical expressions, providing representative coverage across basic algebra, differential and integral calculus, number theory, and linear algebra. A subset of expressions drawn directly from the University of Westminster Mathematics for Computing module 4 slides to evaluate performance against authentic academic content. Additional expressions were included to assess the model's general mathematical reasoning capability.

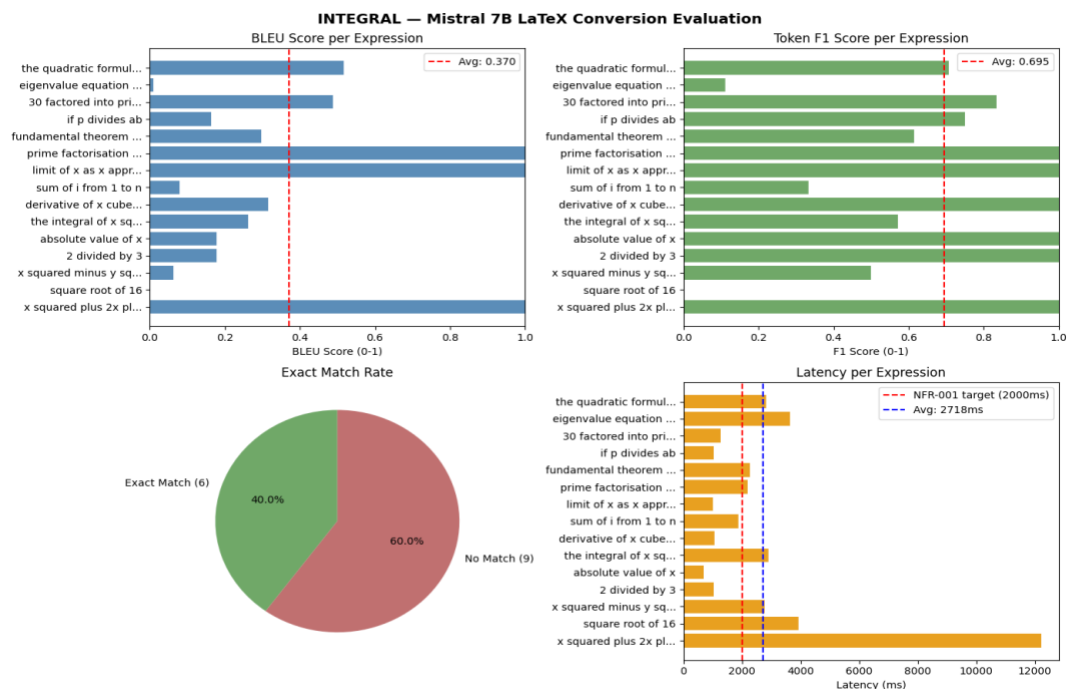


Figure 5.1 Mistral 7B LaTeX Conversion Evaluations

Following metrics were employed:

- BLEU Score (average: 0.370) - a metric originating from the machine translation evaluation that quantifies n-gram overlap between predicted output and reference string, providing a measure of surface-level similarity.
- Token F1 (average: 0.695) - a token-level metric computing the harmonic mean of the precision and recall, affording partial credit to outputs that capture the correct LaTeX tokens irrespective of exact ordering or whitespace.
- Exact Match (40.0%, 6/15 expressions) - the most stringent metric employed, which requires character-perfect correspondence between the predicted and reference LaTeX strings.

As illustrated in Figure 5.1, performance exhibits notable variations across expression categories. Domain-specific conceptual expressions, including 'prime factorisation of n' and 'fundamental theorem of arithmetic', achieved high F1 Token scores following the incorporation of domain-relevant few shot examples into the system prompt. Elementary symbolic expressions such as 'x squared plus 2x plus 1' consistently produced exact matches.

Conversely, lower scores observed for expressions such as 'eigenvalue equation' are attributable to the inherent ambiguity in natural language descriptions of mathematical concepts, where multiple semantically equivalent LaTeX representations exist, thus a known limitation of reference-based evaluation metrics in this domain. In addition, few input expressions which don't correlate to the 'Mathematics for Computing' Module 4 slides of University of Westminster, were purely tested for processing power and logical reasoning, in response to Mistral 7B outputs.

With respect to system latency, the observed average response time of 2.718ms marginally exceeds the NFR-001 performance target of 2,000ms. This deviation is primarily attributable to the computational constraints imposed by executing a 4.4GB parameter model on development machines possessing a RAM of 8GB, leaving limited memory headroom for concurrent processes. It is anticipated that deployment on a dedicated server infrastructure with greater capacity would yield response times comfortably within the specified threshold. Moreover, the latency overhead associated with local model inference is considered an acceptable trade-off given the GDPR compliance requirements of the university deployment context, which preclude the use of lower-latency cloud-based alternatives.

In addition, the system incorporates graceful error handling throughout the conversion pipeline. In the event of model timeout or service unavailability, the system rises typed exceptions such as `ConnectionError`, `TimeoutError`, and `RuntimeError`, which enables the API layer to return structured HTTP error responses of the frontend rather than failing silently. An in-memory cache mechanism further contributes to reliability, returning previously computed results in under 30ms for repeated expressions, thus reducing load on the inference service during high-frequency usage.

Feature	INTEGRAL(Mistral 7B)	MathPix	ChatGPT (GPT-4)
LaTeX conversion	X	X	X
LaTeX deployment	X	0	0
GDPR compliant	X	0	0
Voice input	X	0	0
Cost	Free	Subscription	Per token
Accuracy	Moderate	High	High

Figure 5.2 Comparison Table with Existing Tools

INTEGRAL's primary differentiator is its fully local, privacy-preserving architecture. Existing tools such as Mathpix and ChatGPT offer higher LaTeX conversion accuracy but transmit user data to external servers, rendering them unsuitable for GDPR-sensitive university environments. In addition, INTEGRAL sacrifices some accuracy in exchange for data sovereignty, zero operational cost, and integrated voice dictation, thus providing a combination that's not offered by any existing tool evaluated.

6 Team Work and Collaboration

The INTEGRAL project was delivered using a collaborative Students-as-Co-Creators (SaCC) approach, in which academic staff and students worked as partners throughout the project lifecycle. Rather than following a traditional supervisor-led model, responsibility for planning, implementation, and decision-making was largely delegated to the student team, with academic staff providing guidance, technical advice, and strategic oversight. This approach encouraged greater ownership of the project while developing professional skills in teamwork, communication, and collaborative software development.

Although responsibilities evolved throughout the project, the team initially allocated tasks according to each member's technical expertise and interests. Erik took primary responsibility for frontend development, designing and implementing the user interface, login functionality, and supporting code reviews. Matthew led the backend development, implementing the application architecture, REST API, authentication system, and database integration. Mantas was responsible for the artificial intelligence components of the system, including the integration of the Mistral 7B large language model and Whisper speech recognition pipeline. He also designed and conducted the evaluation of the AI components using quantitative metrics, including BLEU score, F1 score, and exact match scoring, to assess the quality and accuracy of the generated mathematical notation. Roshkan focused primarily on the written report, producing the project overview, defining the project objectives, and completing the critical analysis, risk assessment, and sustainability evaluation. These contributions ensured that the final report presented not only the technical achievements of the project but also a balanced discussion of its limitations, long-term viability, and wider educational impact.

Despite the allocation of primary responsibilities, the project was not completed through isolated individual contributions. Team members regularly supported one another by reviewing code, discussing implementation approaches, identifying software defects, and providing feedback on documentation. This collaborative approach reduced dependency on individual contributors and helped maintain consistency across the different components of the system.

Academic staff acted primarily as mentors rather than project managers. Regular progress meetings provided opportunities to discuss technical challenges, review completed work, and identify priorities for the following development cycle. Rather than prescribing implementation decisions, supervisors encouraged the team to justify design choices and critically evaluate alternative approaches. For example, discussions regarding system latency highlighted that achieving sub-1.5-second response times should be balanced against the advantages of local AI deployment, including improved privacy, compliance with UK GDPR requirements, and reduced dependence on external cloud services. These discussions influenced both the evaluation methodology and the interpretation of the system's performance results, demonstrating how academic guidance contributed to critical thinking rather than directing implementation.

Collaboration during software development was managed using Git and GitHub. Feature branching, pull requests, and peer review formed the core development workflow, allowing individual components to be developed independently before integration into the main codebase. This approach significantly reduced merge conflicts and provided an opportunity for code quality assurance through peer review. However, the project also highlighted the importance of timely reviews. On several occasions, pull requests remained unreviewed for extended periods, delaying integration and preventing comprehensive end-to-end testing. As the project progressed, the team addressed this issue by communicating more proactively through the group messaging platform, ensuring that pending reviews and technical blockers were identified and resolved more quickly.

During the final stages of development, task allocation became increasingly dynamic. Rather than following a rigid project plan, responsibilities were redistributed according to immediate project priorities and remaining deliverables. Matthew coordinated user acceptance testing, maintained the testing register, and supported system integration. Manta completed the performance evaluation of the AI components and conducted benchmarking of the language model. Erik finalised frontend functionality, implemented the search interface, and completed user interface testing. This flexible allocation of work enabled the team to respond effectively to changing priorities while ensuring that all critical project components were completed before submission.

One of the most significant technical challenges encountered throughout development was maintaining a consistent development environment across multiple machines. Differences in local configurations, dependency management, and project directory structures occasionally prevented team members from running the application successfully. For example, the backend required the application to be executed from a specific directory using the command `uvicorn main:app --reload`, a requirement that was not initially documented within the project README. Resolving these issues required close collaboration between team members and highlighted the importance of comprehensive project documentation, reproducible development environments, and clear onboarding procedures for collaborative software projects.

The collaborative review process also proved valuable in identifying software defects before deployment. During the final testing phase, several critical issues were discovered, including an incorrect schema definition that prevented note creation and a missing authentication configuration affecting JSON Web Token generation. Although these issues delayed final integration, they also demonstrated the value of systematic peer review and collaborative testing in improving software reliability and reducing implementation risks.

Reflecting on the project as a whole, one of the most valuable aspects of the Students-as-Co-Creators approach was the emphasis on shared responsibility rather than hierarchical management. While formal project management practices such as daily stand-up meetings were not adopted, the team maintained effective communication through regular meetings, GitHub collaboration, and group messaging. Open discussion of technical challenges, uncertainties, and implementation decisions enabled problems to be addressed before they significantly affected project progress. Equally important, the project provided valuable experience of professional software engineering practices, including collaborative development, version control, peer review, technical communication, and shared decision-making. These experiences not only contributed to the successful delivery of the INTEGRAL prototype but also developed transferable skills directly applicable to future careers in software engineering and artificial intelligence development.

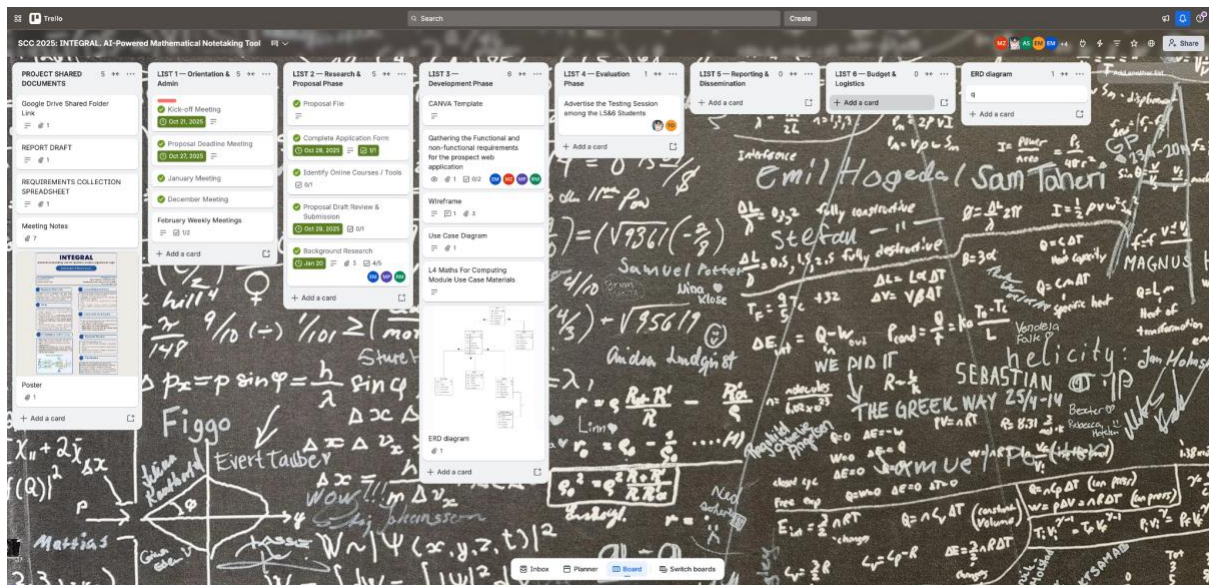


Figure 6.1 Trello board - Task Management Screenshot

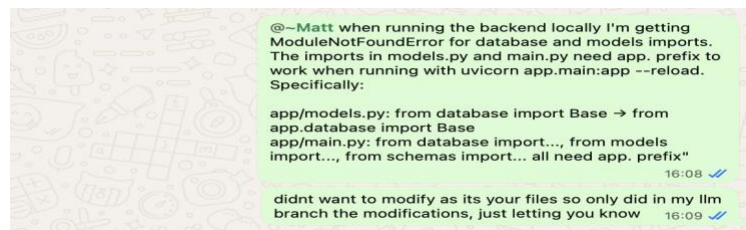


Figure 6.2 Mantas's Messages - WhatsApp Groupchat Screenshot

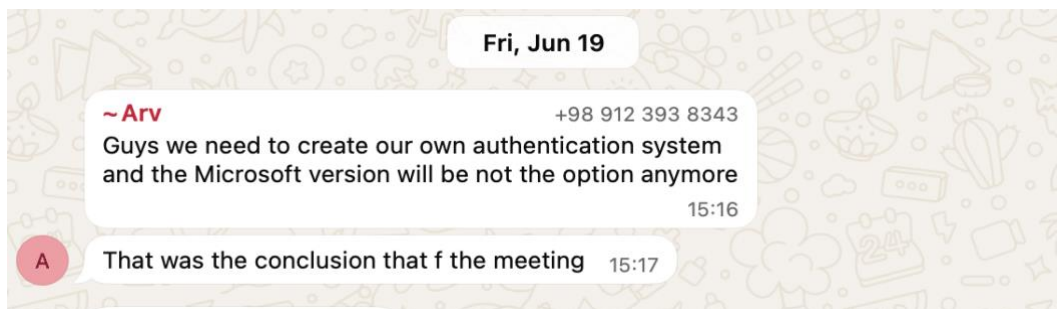


Figure 6.3 Arvin's Messages - WhatsApp Groupchat Screenshot

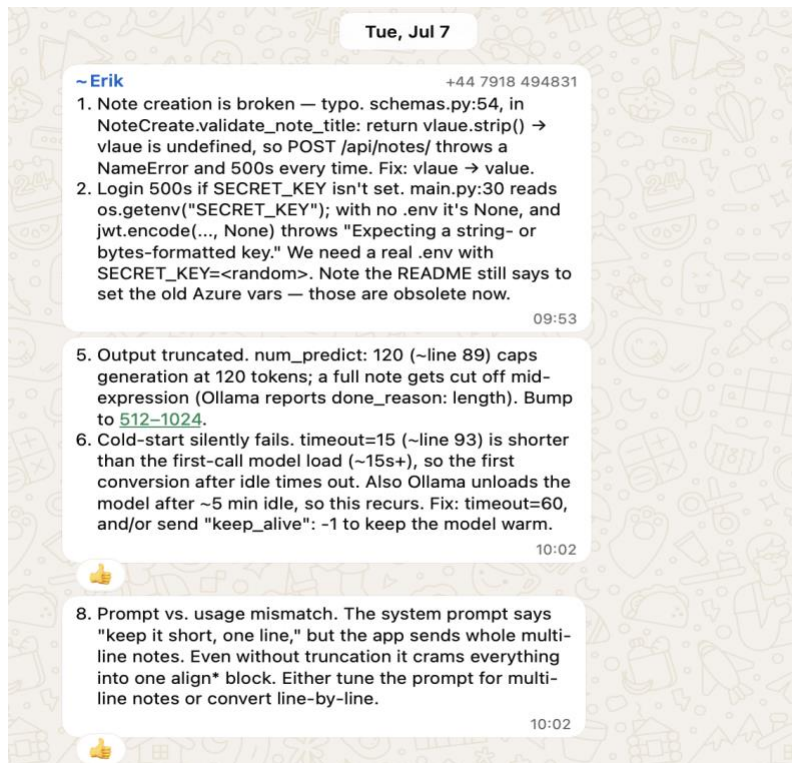


Figure 6.4 Erik's Messages - WhatsApp Groupchat Screenshot

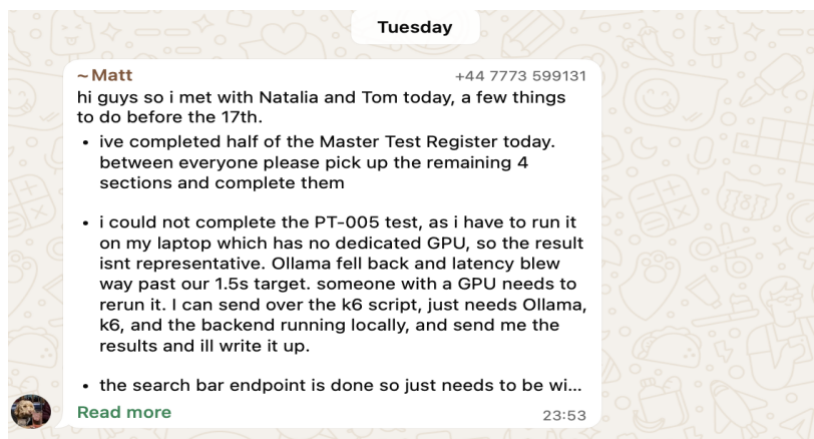


Figure 6.5 Matthew's Messages - WhatsApp Groupchat Screenshot

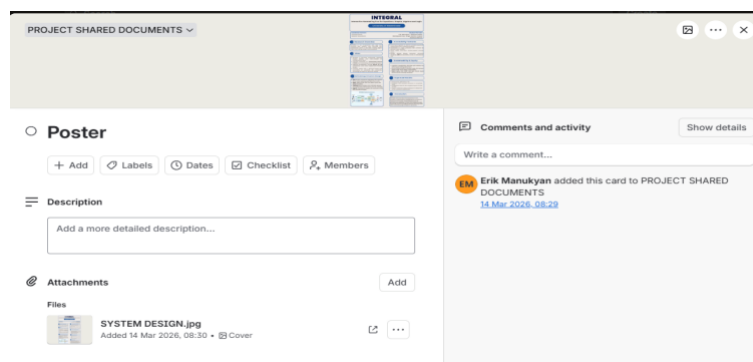


Figure 6.6 Erik's Messages - Trello Workspace Screenshot

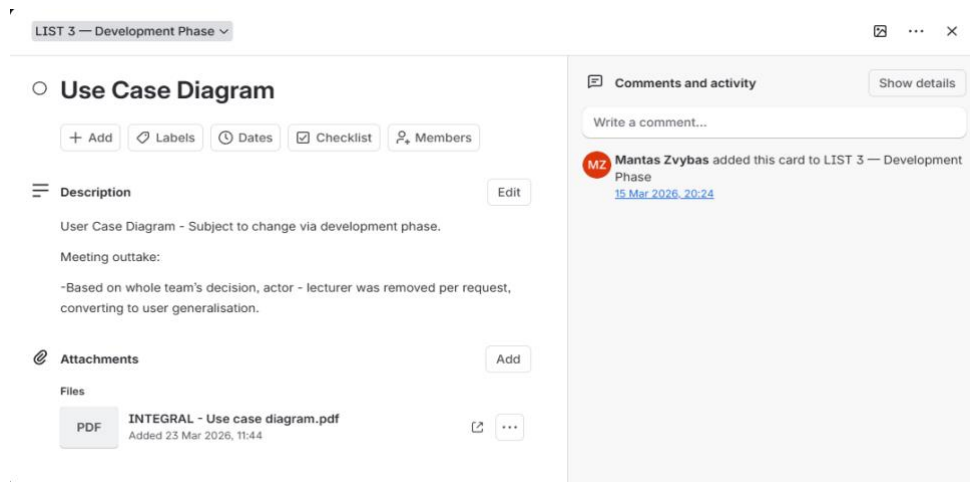


Figure 6.7 Erik's Messages - Trello Workspace Screenshot

Conclusion

The INTEGRAL project set out to test whether generative AI could reduce the representational friction of digital mathematical note-taking without displacing the reasoning it is meant to support. Judged against the six research questions and six objectives set out in the introduction, this aim was substantially achieved.

RQ1 (how AI can improve mathematical note-taking practices) was addressed and empirically demonstrated, if only at prototype scale. The literature review positioned the LLM as a "representational mediation layer" that translates informal input into LaTeX without replacing student reasoning, and the implementation built exactly this: a working text and speech pipeline that UAT-001 and UAT-002 confirmed produces correctly formatted, readable LaTeX from natural language. What the project has not shown is whether this improves note-taking practice over time. The evidence is that the mechanism works, not that it changes how students actually take notes.

RQ2 (what makes digital mathematical note-taking harder than handwriting) was addressed thoroughly, but through literature synthesis rather than original testing. Section 1.1 makes a strong case, grounded in cognitive load theory and the specific demands of mathematical notation (fractions, matrices, superscripts), for why LaTeX's syntactic overhead competes with conceptual understanding. No direct comparative study against handwritten note-taking was run within the project itself, so this answer rests on existing research rather than INTEGRAL's own data.

RQ3 (how AI can enhance usability and accessibility) was partially addressed. The dual text/speech input, zero-install browser design, and configurable theming were all implemented and are consistent with the project's universal-design framing. However, the report's own accessibility discussion flags that LaTeX and image-based notation remain difficult for screen readers to parse, and this was never verified empirically. UI and compatibility testing are recorded as "Not Started" in the Master Test Register. The design intention is sound; the accessibility claim is currently unverified.

RQ4 (how AI-assisted note-taking supports learning, comprehension, and retention) is the least answered of the six. The report leans heavily on constructivist and cognitive load theory to argue that AI should preserve germane cognitive effort, and this theory shaped real design choices, most visibly the deliberate, learner-triggered Convert button rather than automatic conversion. But no learning outcome was actually measured; UAT feedback concerned usability, not comprehension or retention. This question is addressed conceptually, not empirically.

RQ5 (environmental and cognitive implications of paper versus AI-assisted note-taking) was addressed narratively rather than quantitatively. Section 1.4 draws a reasonable comparison between the near-zero energy cost of handwriting and the aggregate energy demands of transformer inference at scale, and local deployment was explicitly chosen to mitigate this. But the project never measured the actual energy consumption of its own system, so the comparison remains a literature-informed argument rather than project-generated evidence.

RQ6 (designing for equitable access across technological contexts) is the most concretely answered research question in the report. The entire local-deployment architecture (no specialist hardware, no per-request cost, browser-based, running on standard university lab machines) is a direct, implemented response to this question, and Figure 5.1.2 makes the equity case explicit by contrasting INTEGRAL's zero cost against MathPix's subscription and ChatGPT's per-token pricing. The main caveat is that "different technological contexts" was tested only within University of Westminster's own lab environment, not against students' personal or lower-spec devices.

(O1) was fully met. The literature review established that LaTeX's syntactic overhead, rather than mathematical understanding itself, is the primary barrier to digital notetaking, and identified AI-mediated translation as an underexplored way to reduce this barrier while preserving learner reasoning.

(O2) was met for its stated scope. Typed and spoken input were both implemented, converged on a single conversion pipeline, and confirmed to work end-to-end in functional and user-acceptance testing (FT-005, UAT-001, UAT-002). The image-based handwriting input originally envisaged in the project's broader ambitions was not delivered within the timeframe and remains future work.

(O3) was met, yielding a subtle conclusion rather than a simple confirmation. Mistral 7B and Whisper proved viable for local, privacy-preserving, cost-free deployment, but evaluation also surfaced real trade-offs: moderate conversion accuracy (40% exact match across the 15-expression test set), and response latency that marginally exceeded the 2-second target on average and degraded further under concurrent load, though without causing failures. This limitation traced to development hardware constraints rather than the underlying architecture.

(O4) was fully met. The three-tier React/FastAPI/SQLite architecture, JWT authentication, and local AI inference layer were all implemented and are functioning, though the originally planned Microsoft SSO integration had to be replaced with a custom authentication system for institutional reasons beyond the team's control.

(O5) was substantially, but not completely, met. Functional, validation, security, and performance testing were all completed with documented pass/fail evidence, and two of three planned user-acceptance scenarios were carried out and surfaced genuine usability issues. UI, compatibility, and integration testing, together with the third UAT scenario, remain outstanding at the time of writing and should be completed before any wider deployment is considered.

(O6) was fully, and arguably most thoroughly, met. The risk, accessibility, ethical, and sustainability analysis was not treated as an afterthought. Rather, they directly shaped the system's central architectural decision to run inference locally rather than through a cloud API.

Taken together, the project delivered a working prototype that demonstrates its central hypothesis is technically feasible: natural-language-to-LaTeX conversion can run locally, privately, and at no marginal cost, while keeping the learner in explicit control of the transformation. What it has not yet demonstrated is whether this translates into improved

learning outcomes; that question sits outside the scope of a system-level evaluation and remains open for future work, alongside the handwriting input mode, server-side token revocation, and the outstanding testing categories identified above. INTEGRAL therefore stands as a credible proof of concept for privacy-preserving, AI-assisted mathematical notetaking, with a clear and well-documented path towards a deployment-ready system.

The most immediate priority is closing the gaps the project's own testing has already exposed rather than opening new fronts: completing the outstanding UI, compatibility, and integration testing, resolving the multi-expression segmentation issue identified independently by a user tester, and implementing server-side token revocation so the current logout vulnerability is properly fixed rather than merely narrowed. Conversion accuracy is the other pressing technical concern, with only 40% exact match and latency exceeding target under load, integrating symbolic verification tools such as SymPy, and re-testing on adequately provisioned hardware rather than an 8GB development machine, would clarify how much of that shortfall is a genuine model limitation versus a hardware artefact.

Beyond fixing what's broken, the more interesting research direction is the question this report identifies but doesn't answer: whether INTEGRAL actually improves learning, not just conversion accuracy. Every evaluation in this project measures system performance (latency, BLEU score, exact match) but none measures cognitive load, retention, or conceptual understanding in real learners over time. A longitudinal study comparing students using INTEGRAL against those using LaTeX directly or pen-and-paper notes would test the project's central pedagogical claim, that AI can absorb representational friction without eroding germane cognitive effort, rather than simply asserting it from theory.

Finally, completing the handwriting input mode would fulfil the multimodal vision set out from the start, and revisiting model scale (comparing 7B against distilled or quantised larger models) could show whether accuracy is improvable without breaching the real-time latency constraints that local deployment was chosen to protect.

References

Anthology (2026) *About Ally and accessibility*. Available at: <https://help.anthology.com/ally-lms/en/about-ally-and-accessibility.html> (Accessed: 10 July 2026).

Arcavi, A. (1994) 'Symbol sense: Informal sense-making in formal mathematics', *For the Learning of Mathematics*, 14(3), pp. 24–35.

Arden, B., Norris, J., Cole, S. and Gamble, S.C. (2024) 'Digital notetaking in lectures: How students adapt to a multi-faceted university learning environment', *Cogent Education*, 11(1), 2373552. doi:10.1080/2331186X.2024.2373552.

Bahls, P. and Wray, A. (2015) 'LaTeXnics: The effect of specialised typesetting software on STEM students' composition processes', *Computers and Composition*, 37, pp. 104–116.

Biggs, J., Tang, C. and Kennedy, G. (2022) *Teaching for quality learning at university*. 5th edn. Maidenhead: Open University Press.

Bolter, J.D. (2001) *Writing space: Computers, hypertext, and the remediation of print*. 2nd edn. Mahwah, NJ: Lawrence Erlbaum Associates.

Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E. et al. (2021) *On the Opportunities and Risks of Foundation Models*. Stanford, CA: Stanford Center for Research on Foundation Models

British Educational Research Association (BERA) (2024) *Ethical guidelines for educational research*. 5th edn. London: BERA.

Brooke, J. (1996) 'SUS: A "quick and dirty" usability scale', in Jordan, P.W., Thomas, B., Weerdmeester, B. and McClelland, I.L. (eds.) *Usability evaluation in industry*. London: Taylor & Francis, pp. 189–194.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020) 'Language models are few-shot learners', *Advances in Neural Information Processing Systems*, 33.

CAST (2024) *Universal Design for Learning Guidelines, Version 3.0*. Available at: <https://udlguidelines.cast.org/> (Accessed: 10 July 2026).

Castelló, M. and Monereo, C. (2005) 'Students' note-taking as a knowledge-construction tool', *L1 – Educational Studies in Language and Literature*, 5(3), pp. 265–285.

Chacon, S. and Straub, B. (2014) *Pro Git*. 2nd edn. New York: Apress.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G. et al. (2021) 'Evaluating large language models trained on code', *arXiv preprint*, arXiv:2107.03374.

Chi, M.T.H. and Wylie, R. (2014) 'The ICAP framework: Linking cognitive engagement to active learning outcomes', *Educational Psychologist*, 49(4), pp. 219–243.

Choi, G.W. and Seo, J. (2024) 'Accessibility, usability, and Universal Design for Learning: Discussion of three key LX/UX elements for inclusive learning design', *TechTrends*, 68(5), pp. 936–945. doi:10.1007/s11528-024-00987-6.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R. *et al.* (2021) 'Training verifiers to solve math word problems', *arXiv preprint*, arXiv:2110.14168.

Dilling, F. and Herrmann, M. (2024) 'Using large language models to support pre-service teachers' mathematical reasoning—An exploratory study on ChatGPT as an instrument for creating mathematical proofs in geometry', *Frontiers in Artificial Intelligence*, 7, 1460337. doi:10.3389/frai.2024.1460337.

Duval, R. (2006) 'A cognitive analysis of problems of comprehension in a learning of mathematics', *Educational Studies in Mathematics*, 61(1–2), pp. 103–131.

European Commission (2019) *Ethics guidelines for trustworthy AI*. Brussels: European Commission High-Level Expert Group on Artificial Intelligence.

FastAPI (2025) *FastAPI documentation*. Available at: <https://fastapi.tiangolo.com/> (Accessed: 25 July 2026).

Fernandez, J., Na, C., Tiwari, V., Bisk, Y., Luccioni, S. and Strubell, E. (2025) *Energy considerations of large language model inference and efficiency optimizations*. Available at: <https://arxiv.org/abs/2504.17674> (Accessed: 10 July 2026).

Friedman, M.C. (2014) *Notes on note-taking: Review of research and insights for students and instructors*. Cambridge, MA: Harvard Initiative for Learning and Teaching.

Gkritzali, E., Kaliosis, P., Galanaki, S., Palogiannidi, E. and Giannakopoulos, T. (2024) 'Greek2MathTex: A Greek speech-to-text framework for LaTeX equations generation', in *Proceedings of the 13th Hellenic Conference on Artificial Intelligence (SETN '24)*. New York: ACM, Article 19, pp. 1–4. doi:10.1145/3688671.3688771.

Grätzer, G. (2008) 'A gentle learning curve for LaTeX', *TUGboat / PracTeX Journal*.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. and Steinhardt, J. (2021) 'Measuring massive multitask language understanding', *Proceedings of the International Conference on Learning Representations (ICLR 2021)*.

Hiebert, J. and Lefevre, P. (2013) 'Conceptual and procedural knowledge in mathematics: An introductory analysis', in Hiebert, J. (ed.) *Conceptual and procedural knowledge*. London: Routledge, pp. 1–27.

Holmes, W., Bialik, M. and Fadel, C. (2019) *Artificial intelligence in education: Promises and implications for teaching and learning*. Boston, MA: Center for Curriculum Redesign.

Husom, E.J., Goknil, A., Astekin, M., Shar, L.K., Kåsen, A., Sen, S., Mithassel, B.A. and Soylu, A. (2025) *Sustainable LLM inference for edge AI: Evaluating quantized LLMs for energy efficiency, output accuracy, and inference latency*. Available at: <https://arxiv.org/abs/2504.03360> (Accessed: 7 July 2026).

Information Commissioner's Office (2024) *Guide to UK GDPR*. Available at: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/> (Accessed: 25 July 2026).

International Organization for Standardization (ISO) (2018) *ISO 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. Geneva: International Organization for Standardization.

International Organization for Standardization (ISO) (2019) *ISO 9241-210:2019 Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. Geneva: International Organization for Standardization.

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L., Lachaux, M.A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T. and Sayed, W. (2023) *Mistral 7B*. Available at: <https://arxiv.org/abs/2310.06825> (Accessed: 10 July 2026).

Jurafsky, D. and Martin, J.H. (2025) *Speech and Language Processing*. 3rd edn. Draft. Available at: <https://web.stanford.edu/~jurafsky/slp3/>

Kapur, M. (2014) 'Productive failure in learning mathematics', *Cognitive Science*, 38(5), pp. 1008–1022. doi:10.1111/cogs.12107.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeiffer, F., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M. and Kasneci, G. (2023) 'ChatGPT for good? On opportunities and challenges of large language models for education', *Learning and Individual Differences*, 103, 102274.

KaTeX Contributors (2025) *KaTeX documentation*. Available at: <https://katex.org/> (Accessed: 25 July 2026).

Kirsch, D. (2010) *Detexify: LaTeX handwritten symbol recognition* [Software]. Available at: <https://detexify.kirelabs.org> (Accessed: 10 July 2026).

Knauff, M. and Nejasmic, J. (2014) 'An efficiency comparison of document preparation systems used in academic research and development', *PLoS ONE*, 9(12), e115069.

Kortemeyer, G. (2023) 'Using artificial intelligence tools to make LaTeX content accessible to blind readers', *TUGboat*, 44(3), pp. 392–399.

Korzh, D., Shapovalov, Y., Kramarenko, O. et al. (2025) *Speech-to-LaTeX: New models and datasets for converting spoken equations and sentences*. Available at: <https://arxiv.org/abs/2508.03542> (Accessed: 10 July 2026).

Lamport, L. (1994) *LaTeX: A document preparation system: User's guide and reference manual*. 2nd edn. Reading, MA: Addison-Wesley.

Maćkowski, M., Kawulok, M., Brzoza, P. and Spinczyk, D. (2023) 'Methods and tools supporting the learning and teaching of mathematics dedicated to students with blindness', *Applied Sciences*, 13(12), 7240.

Manzoor, A., Shafait, F. et al. (2019) 'ALAP: Accessible LaTeX-based mathematical document authoring and presentation', in *Proceedings of the CHI Conference on Human Factors in Computing Systems*. New York: ACM.

Martin, A.J., Collie, R.J. et al. (2025) 'Integrating generative AI and load reduction instruction to individualise and optimise students' learning', *Learning and Individual Differences*, 121, 102723.

Mayer, R.E. and Moreno, R. (2003) 'Nine ways to reduce cognitive load in multimedia learning', *Educational Psychologist*, 38(1), pp. 43–52.

Microsoft (2025a) *TypeScript documentation*. Available at: <https://www.typescriptlang.org/docs/> (Accessed: 25 July 2026).

Microsoft (2025b) *Visual Studio Code documentation*. Available at: <https://code.visualstudio.com/docs> (Accessed: 25 July 2026).

Mueller, P.A. and Oppenheimer, D.M. (2014) 'The pen is mightier than the keyboard: Advantages of longhand over laptop note taking', *Psychological Science*, 25(6), pp. 1159–1168.

Nielsen, J. (1994) *Usability engineering*. San Francisco: Morgan Kaufmann.

Norman, D.A. (2013) *The design of everyday things*. Revised and expanded edn. New York: Basic Books.

Ollama (2025) *Ollama documentation*. Available at: <https://ollama.com/> (Accessed: 25 July 2026).

OpenAI (2023) *GPT-4 technical report*. Available at: <https://arxiv.org/abs/2303.08774> (Accessed: 10 July 2026).

OpenAI (2025) *OpenAI API Documentation*. Available at: <https://platform.openai.com/docs> (Accessed: 31 July 2026)

Pressman, R.S. and Maxim, B.R. (2020) *Software engineering: A practitioner's approach*. 9th edn. New York: McGraw-Hill.

Quinby, F., Pollanen, M., Reynolds, M.G. and Burr, W.S. (2020) 'Effects of digitally typesetting mathematics on working memory', in *Engineering Psychology and Cognitive Ergonomics*. Cham: Springer.

Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C. and Sutskever, I. (2022) *Robust speech recognition via large-scale weak supervision*. Available at: <https://arxiv.org/abs/2212.04356> (Accessed: 10 July 2026).

React (2025) *React documentation*. Available at: <https://react.dev/> (Accessed: 25 July 2026).

Risko, E.F. and Gilbert, S.J. (2016) 'Cognitive offloading', *Trends in Cognitive Sciences*, 20(9), pp. 676–688.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

Sánchez Egea, A.J. [et al.]. Cloud-based LaTeX editors for collaborative academic projects, case study in Overleaf. A: Proceedings of International EPS Conference (5th International EPS Conference). Vilanova i la Geltrú (Barcelona), 23-25 October 2024. Vilanova i la Geltrú. Universitat Politècnica de Catalunya, 2026.

Schoenfeld, A.H. (2014) 'What makes for powerful classrooms, and how can we support teachers in creating them?', *Educational Researcher*, 43(8), pp. 404–412.

Sommerville, I. (2020) *Software engineering*. 10th edn. Harlow: Pearson.

Sparrow, B., Liu, J. and Wegner, D.M. (2011) 'Google effects on memory: Cognitive consequences of having information at our fingertips', *Science*, 333(6043), pp. 776–778.

Sweller, J. (1988) 'Cognitive load during problem solving: Effects on learning', *Cognitive Science*, 12(2), pp. 257–285.

Sweller, J., van Merriënboer, J.J.G. and Paas, F.G.W.C. (1998) 'Cognitive architecture and instructional design', *Educational Psychology Review*, 10(3), pp. 251–296.

Tailwind Labs (2025) *Tailwind CSS documentation*. Available at: <https://tailwindcss.com/docs> (Accessed: 25 July 2026).

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023) *Llama 2: Open foundation and fine-tuned chat models*. Available at: <https://arxiv.org/abs/2307.09288> (Accessed: 10 July 2026).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) 'Attention is all you need', *Advances in Neural Information Processing Systems*, 30.

Wang, X., Wang, J., Li, Y. et al. (2024) *Large language models for education: A survey and outlook*. Available at: <https://arxiv.org/abs/2403.18105> (Accessed: 10 July 2026).

Wittrock, M.C. (1992) 'Generative learning processes of the brain', *Educational Psychologist*, 27(4), pp. 531–541.

World Wide Web Consortium (W3C) (2023) *Web Content Accessibility Guidelines (WCAG) 2.2*. Available at: <https://www.w3.org/TR/WCAG22/> (Accessed: 25 July 2026).

Zhang, H. and Jiang, R. (2026) 'Cognitive load pathways linking generative AI dependency to university students' higher-order thinking skills and academic enjoyment', *Education and Information Technologies*. doi:10.1007/s10639-026-14026-9.

Supporting Materials and Appendices

This chapter provides an overview of the supplementary materials accompanying the report. The appendices support the project's research, system design, implementation, evaluation, dissemination activities, and technical development.

Appendix 1: Prospective User Questionnaire

Appendix 1 contains the INTEGRAL prospective user questionnaire, developed to investigate current mathematical note-taking practices, common challenges, accessibility requirements, and desired system features.

The questionnaire explores participants' note-taking methods, difficulties with mathematical notation, equations and graphs, familiarity with LaTeX, preferred accessibility settings, and the importance of features such as AI-assisted LaTeX generation, graph rendering, note organisation, export functionality, and speech input. Participants were also asked how likely they would be to use INTEGRAL during lectures, revision, coursework, examination preparation, and research.

Appendix 2: Requirements Collection Spreadsheet

Appendix 2 contains the complete Software Requirements Specification (SRS) developed during the requirements elicitation phase of the INTEGRAL project. The spreadsheet documents the functional and non-functional requirements identified through the prospective user questionnaire, literature review, and discussions within the Students-as-Co-Creators project team. Each requirement is recorded using a structured format that includes a unique identifier, description, user story, acceptance criteria, implementation priority (MoSCoW), quality attributes, target metrics, associated system components, implementation status, and supporting notes. This document provided the primary reference throughout the project's design and development, ensuring traceability from identified user needs to the implemented system features and evaluation criteria.

Appendix 3: Conference Poster and Abstract

Appendix 3 contains the poster and abstract presented at the *9th Student–Staff Research Conference*, held at the University of Westminster on 25 February 2026. These materials summarise the project's motivation, objectives, and proposed solution, providing evidence of the dissemination of the project to the University's research community.

Appendix 4: Testing and Evaluation Tables

Appendix 4 presents the complete testing and evaluation documentation. It includes detailed test cases, expected and actual outcomes, identified issues, implemented fixes, and the final status of each test.

These materials provide supporting evidence for the evaluation presented in Chapter 5 and demonstrate the systematic assessment of the system's functionality, usability, security, performance, and reliability.

Appendix 5: Workshop Presentation and Promotional Materials

Appendix 5 contains the presentation slides and promotional materials developed for the *INTEGRAL: AI for Mathematical Notetaking Workshop*, held at the University of Westminster on 7 April 2026. These materials were used to introduce the project, demonstrate the prototype, and engage students and staff with the project's aims and outcomes.

Appendix 6: GitHub Repository

Appendix 6 provides access to the project's GitHub repository, which contains the complete source code, implementation files, supporting scripts, technical documentation, and development resources created throughout the project.

The repository also documents the evolution of the system through version control, demonstrating the technical contributions of the project team and providing evidence of the iterative software development process.