2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC

Host Sponsor UNC Charlotte

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #53256

Tooling for digital accessibility in mathematics: Quickly build compliant course websites that benefit all students

Dr. Matthew McMillan, University of Virginia

Matthew McMillan is an assistant professor in the Center for Applied Mathematics at the University of Virginia. Before joining UVA, Matthew was a PhD student in mathematics at UCLA, where his research focused on higher representation theory. Matthew also has a background in experimental and computational plasma physics, the latter involving mathematical modeling and simulation in stellarators (fusion reactors). Besides pure math and math pedagogy, Matthew maintains scholarly interest in the psychology, history, and philosophy of math.

Eli Daniel Boyden, University of Virginia

Eli Boyden is an undergraduate at UVA studying Computer Science and Mathematics. He has experience as a teaching assistant in applied mathematics and has worked in software engineering roles spanning frontend and backend systems.

Tooling for digital accessibility in mathematics: Quickly build compliant course websites that benefit all students

Abstract

Public universities in the US are now required to meet digital accessibility (DA) standards under the 2024 updates to Title II of the Americans with Disabilities Act (ADA). For mathematics instructors, this means course materials must be parsable by screen readers, but conventional LaTeX-to-PDF workflows historically do not provide such materials. Despite the availability of Mathematical Markup Language (MathML) as a web standard for accessible math content, supported by all modern browsers, instructor adoption of DA-compliant materials remains very low. This creates a gap between available technology and classroom practice regarding compliance with digital accessibility standards.

This paper makes three contributions toward closing this gap. First, we present a taxonomy of existing approaches for DA-compliant math content, organized by whether they are optimized for print (PDF) or web (HTML) outputs, and we analyze their respective tradeoffs for instructor adoption. Second, we describe and document a proposed implementation: a free software workflow using Obsidian (Markdown-based document authoring and content management tool), Quartz (static site generator), Git (collaboration and version control), and Cloudflare Pages (build and host static sites). This workflow enables math instructors to create, manage, and publish DA-compliant course websites offering MathML from documents that encode math expressions in a TeX-based syntax. The system requires a one-time setup of approximately 1-2 hours, and thereafter site updates occur in minutes, in the cloud, after content authors run a single command. A setup tutorial has been made publicly available by the authors. Third, we present an empirical study of student outcomes across 31 sections of Calculus II over 6 semesters. Student performance data shows that sections using the proposed course website system outperformed control sections, with the treatment group reaching 2.46 standard deviations above the control mean in the final semester of the study. Although all treatment sections were taught by a single instructor, several lines of evidence, such as the acclimation trajectories of other new instructors on the same course, indicate that the system itself may be a meaningful contributor to the observed performance gains. A survey of student experience shows no statistically significant difference between groups, suggesting that the system does not negatively affect student experience. A proposed second phase of the study will assess barriers to instructor adoption at other institutions.

Introduction

Motivation

On April 24th, 2024 the Department of Justice (DOJ) released its final updates for Title II of the Americans with Disabilities Act (ADA), focusing on ensuring accessible web content in state and local government bodies including public universities and private universities that receive federal funding.[1] The technical requirements for this act are dictated by Web Content Accessibility Guidelines (WCAG) 2.1 AA standards produced by the World Wide Web Consortium (W3C). For large public universities, the deadline for compliance is April 2027.

Math instructors are primarily affected by WCAG standard 1.1.1 which requires text alternatives for non-text content.[2] W3C recommends Mathematical Markup Language (MathML) for accessible math content, citing its ability to be parsed into audio, braille, and other output formats.[3] The National Center on Accessible Digital Educational Materials and Instruction likewise recommends MathML for its compatibility with screen readers.[4] Modern screen readers such as Fusion and Job Access With Speech (JAWS) support MathML parsing with audio and braille output.[5] Empirical evidence also suggests that MathML can improve student and teacher experience compared with traditional textbooks.[6]

Despite these standards, recommendations, and technologies, adoption of accessible math content remains low. A survey of 139 physics websites at US post-secondary institutions found accessibility errors on every site.[7] Similar conclusions have been reached regarding digital accessibility in computer science education.[8] MathML adoption specifically has been hampered by historically inconsistent browser support, leading many web developers to offer math notation as rasterized images rather than structured markup.[9] As of 2026, however, MathML is fully supported in every major browser,[10] so the remaining barrier is not technological but practical: instructors lack clear, efficient, and well-documented workflows for producing and serving MathML-based course materials.

This paper addresses that practical barrier. We propose and study a complete system in which instructors author and manage their content pages with familiar TeX-based syntax, publish to a website serving MathML, and rebuild the site after content updates by running a single command.

Contributions

This paper makes three primary contributions:

1. **Taxonomy of approaches to DA compliance.** We present a structured comparison of existing methods for producing accessible math content, organized by whether they target print (Type P) or web (Type W) output, with comments on their tradeoffs for instructor adoption. This taxonomy provides a framework for discussing and evaluating current and future solutions.
2. **A documented, easy-to-use workflow.** We describe a specific Type W implementation based on Obsidian and Quartz, and we provide a step-by-step tutorial designed for instructors with no web development background. We report on the results of using this

system over three semesters.
3. **Empirical study of student outcomes.** We present student performance data and student experience survey results from 31 sections of Calculus II over 6 semesters, comparing sections that used the proposed system with those that did not. Although the experimental design has important limitations (most notably that all treatment sections were taught by a single instructor), within-study comparisons help to isolate the contribution of the system from confounding factors.

A proposed second phase, summarized at the end of this article, will study barriers to adoption of this solution by instructors at other institutions.

Design requirements

To frame the comparison of existing approaches and motivate our design choices, we identify the following theoretical desiderata for DA compliance solutions that could realistically achieve broad adoption among math instructors. A prerequisite for any solution considered here is MathML output: the system must be able to produce MathML embedded in HTML, the recognized standard for accessible math on the web. Since every solution type discussed in this article can produce MathML, we treat it as a baseline capability rather than a distinguishing desideratum, and we compare solutions against the following criteria:

1. *Familiar syntax.* Authors can write math using TeX-based syntax, minimizing retraining.
2. *Low-friction revision.* Instructors frequently update course materials mid-semester, so the edit-to-published cycle must be simple and fast.
3. *Minimal technical prerequisites.* The system should not require command-line interactions after the initial setup, nor any significant web-development expertise.
4. *Free or very low cost.* Budget constraints should not be a barrier to adoption.
5. *Access management.* Authors should have control over (i) what content is published, and (ii) who can access the site.
6. *Content management and possession.* The system should support "Content Management System" (CMS) features useful for courseware: question banks, transclusion (embed-ability), batch publishing (e.g. for homework solutions), links between pages. Authors should also retain possession of their content as portable source files.
7. *AI workflow friendly.* Content source should reside in the instructor's possession in plain-text formats (e.g. Markdown or LaTeX) that integrate readily with AI-assisted workflows for drafting, revising, converting, and batch-editing materials.

These requirements guide the taxonomy discussion and the design of our setup. Note that some of these features go beyond the basic requirements for a successful DA-compliance solution, but they make a system more appealing to instructors and they distinguish our system from certain alternatives.

Related Work and Taxonomy of Solutions

Prior scholarship

Several research efforts have addressed the problem of accessible math content in scholarly articles. Custom systems proposed in the literature include a multimedia platform using a C++ compiler to parse a specialized TeX-like syntax and generate PHP/JavaScript pages,[11] but resources are not publicly available for other instructors to set up and use this system in their own classrooms. Another system uses Python's SymPy library to convert math input to MathML, parses the MathML into semantic components, and offers interactions allowing users to navigate the mathematical structure of the expressions through a keyboard and speech feedback.[12] Other notable software tools addressed in research articles include MathPlayer, which provides MathML browser support now common in modern browsers;[13] MATHTYPE, an editor for MathML and LaTeX content that integrates with other software tools like Word or Canvas;[14] and EuroMath, a web-based platform for mathematical communication that appears to be no longer supported.[15] Recent scholarship has also explored improving PDF accessibility through Markdown conversion using transformer models[16] and by providing supplementary accessible files alongside PDFs.[17]

Taxonomy of solutions

Our project involves one of various possible approaches to DA-compliance. To contextualize our design choices, we present a taxonomy of the solution space according to the output format primarily targeted, along with examples of software implementations in these types.

The example lists are far from exhaustive. Moreover, there are other solution types that we do not discuss, although they may serve well in individual cases. For example, Microsoft Word documents allow authors to enter equations parsable by screen readers directly from the .docx file. For another example, MathCad is a proprietary system for documentation of engineering calculations that allows users to generate HTML+MathML pages representing the users' notebooks. (SMath is a comparable system with a free version.) However, most mathematics professors do not use these systems because they are already accustomed to the TeX-based syntax which is ubiquitous in mathematics (see Requirement 1), and which is more efficient for writing in large volume than palette-based editors such as in Word or MathCad.

The typesetting and rendering procedures we discuss may be divided according to whether they target PDF or HTML as the primary output. The PDF specification is designed for print: it provides fixed-layout pages well suited to publication for the purposes of reference and archiving, and the LaTeX-to-PDF workflow is deeply established among mathematicians. However, educators have different needs from researchers. They update and rewrite materials frequently, they benefit from modular, internally connected, and easily navigable content, and they serve students who read on devices of various sizes. HTML is a digital-first specification designed for content that is interconnected, reflowable, and dynamically maintained, making it intrinsically better suited for course materials. From the standpoint of digital accessibility, PDF files containing math historically have not been usable by screen readers, whereas HTML with embedded MathML is the recognized accessible format.

The landscape is rapidly evolving. Several significant innovation milestones have been reached in the area of accessible PDFs since this project began and this article was drafted and reviewed. In March of 2026, the international PDF Association announced new standards for accessible math in PDF files in the new PDF/UA-2.[18] Efforts are ongoing to implement updates to LaTeX to allow for generation of PDFs containing accessible math.[19] At the same time, there is a growing capacity to make use of these PDFs: although most browsers and PDF readers cannot yet handle embedded math, NVDA and JAWS are now able to do so in Firefox. Converting an old LaTeX file is not necessarily trivial, though, since very many LaTeX packages are partially or completely incompatible with current methods of compiling to the new PDF/UA-2. (See the package compatibility status page.[20]) In any case, instructors desiring a PDF-first approach for their course materials, after considering the other advantages of targeting MathML as described in this paper, are probably well-served to focus on that conversion and the requisite adoption of LuaLaTeX. It is true, of course, that many LaTeX packages are also incompatible with MathJax and KaTeX; our proposal does not aim to maximize compatibility with existing materials.

We denote by "Type P" those systems designed for printable PDF output, and by "Type W" those designed for web HTML output. For DA-compliance, Type P systems preserve existing LaTeX workflows but require post-conversion to HTML, which introduces friction into the revision cycle. Type W systems target HTML directly, offering better revision efficiency and higher quality web output, but requiring authors to adopt new (if simpler) writing tools.

It is important to note that DA regulations require the primary learning resource to be accessible. Offering a lower-quality version of content in an accessible form to a student with accessibility needs, while the other students are offered a higher-quality (primary) version in an inaccessible medium, would not be compliant with DA regulations. A system designed to produce PDFs may suffer from "PDF primacy temptation," which is the result of authors focusing on the PDF output quality for most students, while neglecting the quality of an "accessory" output that is produced only for compliance purposes. Systems designed for PDF output will tend to contribute DA compliance risk in this way.

The following taxonomy further divides Type P systems into P1 and P2, and Type W systems into W1, W2, and W3:

P1 — Machine conversion: PDF → MathML

Method: Software tools, typically implementing ML character recognition, possibly with LLM support, convert the PDF file itself to HTML webpages with embedded MathML. These files are then loaded in the public pages of a course website. Attention is required for links between files.

Example implementations: MathPix (math-aware OCR); EditTrans (LLM-based), PDFix

Pros	Cons
• Meets desiderata 1, 3, 4, 7 • Preserve document shape and layout • Continue using LaTeX source • Same method applies to old materials	• Fails desiderata 2, 5, 6 • Poor author control over resulting HTML • Output HTML requires manual cleanup • Inefficient cycle for continuous revision • DA risk: PDF primacy temptation

P2 — Machine conversion: LaTeX → MathML

Method: Software tools convert LaTeX source code to HTML with MathML.

Example implementations: LaTeXML (used by arXiv.org); Pandoc; tex4ht; Math-Core

Pros	Cons
• Meets desiderata 1, 3, 4, 7 • Preserve document structure • Continue using LaTeX source • Same method applies to old materials	• Fails desiderata 2, 5, 6 • Moderate control over resulting HTML • Must adjust syntax, settings each file • Conversion time discourages frequent revision • DA risk: PDF primacy temptation

W1 — Create and manage content in an LMS

Method: Learning Management Systems like Canvas or Blackboard allow instructors to add math expressions to internal content pages using a built-in equation editor, producing output compatible with built-in accessibility software functions.

Example implementations: Canvas, Blackboard

Pros	Cons
• Meets desiderata 2, 3, 4, 5 • Some limited CMS functionality • Good control over HTML (less over MathML) • Institutional site access control is built in	• Fails desiderata 1, 6, 7 • Layout and TeX limitations (vs. LaTeX) • Poor math editor UI, inadequate for heavy use • Cannot export math source • Cannot export PDF versions • Doesn't apply to old materials

W2 — Create and manage Markdown+TeX source in pro web CMS

Method: Web publishing systems such as Wordpress or Drupal can serve math using HTML including MathML. However, these systems are designed primarily for general web publishing

rather than courseware authoring. They require substantial web development skills and resources to maintain, and they carry significant subscription or licensing costs.

Example implementations: Wordpress, Drupal, SquareSpace, WIX

Pros	Cons
• Meets desiderata 1, 5, 6 • Myriad website features are possible	• Fails desiderata 2, 3, 4, 7 • Some layout and TeX limitations (vs. LaTeX) • Requires web dev skills: setup and ongoing • CMS features for courseware aren't bundled

W3 — Create and manage Markdown+TeX source in personal local CMS

Method: Write course material in local files using Markdown and a TeX variant. Use a static site generator tool to produce HTML with MathML.

Example implementations: Obsidian, LogSeq, Jupyter Notebooks, Hugo, Notion, Craft; Jamstack generators

Pros	Cons
• Meets desiderata 1, 2, 3, 4, 5, 6, 7 • Easy set up compared with pro CMS • Expedite authoring with plugins • CMS functionality fits courseware needs	• Fails desiderata [none] • Some layout and TeX limitations (vs. LaTeX) • PDF output is secondary, lower quality

Summary

System	Primary Strength	Primary Weaknesses
P1	Simplest DA solution	Low quality HTML output; opaque and slow conversion discourages revision
P2	Robust LaTeX-first	Setup time per file; opaque and slow conversion; "PDF primacy" temptation
W1	Easy start	Slow to use; content trapped; no PDFs
W2	Myriad website features possible	Prohibitive prereq skills
W3	Web-first yet serves PDF; easy to write; compelling CMS features	Mildly technical one-time setup; reduced PDF layout control vs. LaTeX

Evaluated against the design requirements identified in the introduction, Type W3 systems best balance the competing demands of accessibility compliance, low to moderate startup overhead, content management functionality that is useful for courseware, and revision cycle efficiency.

Table 1 summarizes this evaluation as a scorecard of the five system types against the numbered design requirements. Our implementation provided below falls into the W3 category.

Design requirement	**P1** PDF→ML	**P2** LaTeX→ML	**W1** LMS	**W2** Pro CMS	**W3** Ours
1. Familiar TeX syntax	✓	✓	✗	✓	✓
2. Low-friction revision	✗	✗	✓	✗	✓
3. Low expertise	✓	✓	✓	✗	✓
4. Low cost	✓	✓	✓	✗	✓
5. Access management	✗	✗	✓	✓	✓
6. Content management and possession	✗	✗	✗	✓	✓
7. AI workflows friendly	✓	✓	✗	✗	✓

Table 1: Scorecard of MathML solution types against the design requirements. The numbered rows correspond to the distinguishing desiderata listed in the introduction.

System Overview

This section describes the system we have implemented. We first present the daily instructor workflow to convey the simplicity of ongoing use, then describe the one-time setup process. For a complete step-by-step tutorial designed for instructors with no web development background, see our public tutorial at `quartz-mathml-tutorial.pages.dev`. Initial setup is expected to take 1-2 hours following the tutorial; after setup, authors work entirely within Obsidian to write content and initiate site rebuilds. The site rebuild is triggered by a command in Obsidian and takes place in the cloud. It requires 2-5 minutes to build our site which hosts all material for six courses.

Daily workflow

Content is written in Obsidian, a highly extensible note-taking application that views and manages a folder, referred to as a "vault," containing Markdown files stored locally on the instructor's computer. Math is written in a TeX-like syntax (with single $ for inline math, double $$ for display math). Obsidian provides a streamlined writing interface with features such as LaTeX autocompletion, internal linking between pages, and transclusion (embedding a part of one page's contents inside another page). These features enable modular courseware: instructors can maintain a personal library of theory notes, examples, and exercises, transclude them into semester-specific pages, manage question banks with linked solutions, and batch-publish homework solutions by toggling a "property" name in the file frontmatter.

Publishing follows the "commit and sync" command provided by the Obsidian Git plugin. This command automatically synchronizes the local folder with the cloud GitHub repository using a "commit" to save and label the current state. Cloudflare Pages detects the update and automatically rebuilds the website in the cloud. The author does not need to take any further steps, and the new site will be live in a few minutes.

One-time setup

The system uses four applications (see Figure 1):

1. **Obsidian** — Composition and content management. Obsidian's simple Markdown file format, internal link management, and vast free plugin ecosystem (including *Obsidian Git* for version control) make it well-adapted for courseware managed by individual math instructors. Although a setup without plugins (besides *Obsidian Git*) would suffice, we recommend considering use of the following plugins while optimizing a workflow over time: *Admonition*, *Advanced Cursors*, *Advanced Tables*, *Better Export PDF*, *Copy Block Link*, *Easy Bake*, *Excalidraw*, *File Cooker*, *Git*, *Linter*, *Multi Properties*, *Note Refactor*, *PDF Break Page*, *Quick Latex* (or *Latex Suite*), *Quick Switcher++*, *Recent Files*, *Settings Search*, *Style Settings*, *Table of Contents*, *Tag Wrangler*, *Text Format*, *Text Transporter*, *TikZJax*, *Underline*.
2. **Git/GitHub** — Version control and cloud storage. Content files are synced with a GitHub repository, enabling multi-instructor collaboration and saved snapshots, and providing the connection point for automated deployment of the site.
3. **Quartz** — Static site generator tool. Quartz is designed and maintained by Jacky Zhao (`jzhao.xyz`) specifically to build websites from Markdown collections in Obsidian or similar apps. It transforms Markdown files into a collection of HTML, CSS, and JavaScript files. Critically, Quartz supports MathML generation with minor customization that we explain in the tutorial. The simplest method just passes an option to the KaTeX parsing engine. We also developed a custom Quartz plugin using the Temml library (via the open-source rehype-mathml project); this provides wider TeX syntax coverage and some MathML rendering improvements. Note that Quartz requires NodeJS as a dependency.
4. **Cloudflare Pages** — Free web hosting with automatic build and deployment triggered by changes to the GitHub repository. Other hosting options (Vercel, Netlify, self-hosting) are also possible.

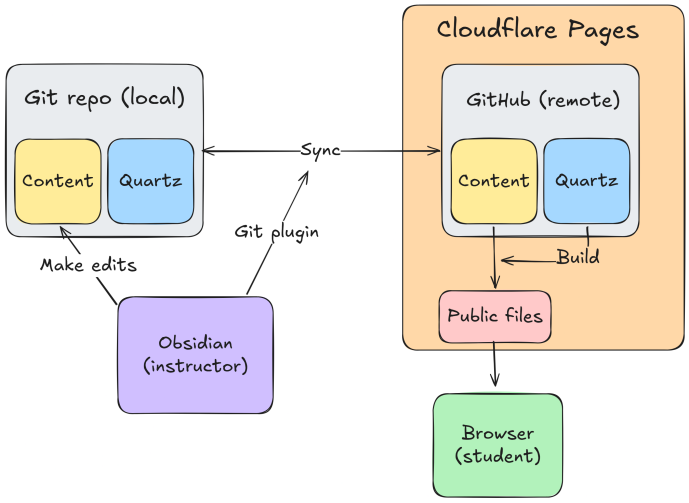

Figure 1: System workflow. The instructor interacts only with Obsidian (left). A single "Commit and Sync" command triggers automatic building and deployment of the course website (right).

Customization is possible at each layer. For example, Quartz supports custom SCSS for styling; we use this to create custom "callout" block styling for concept definitions, examples, exercises, and solutions as are commonly included in math course materials. These customizations and all other setup details are covered in the tutorial.

Documentation links and website screenshots

See Figure 2 for a sample of the course homepage, and Figure 3 for a sample of a homework assignment. Documentation for each software tool is currently found at the following pages:

Obsidian: https://help.obsidian.md/
Git: https://git-scm.com/docs
GitHub: https://docs.github.com/en
Quartz: https://quartz.jzhao.xyz/
NodeJS: https://nodejs.org/docs/latest/api/
Cloudflare: https://developers.cloudflare.com/directory/

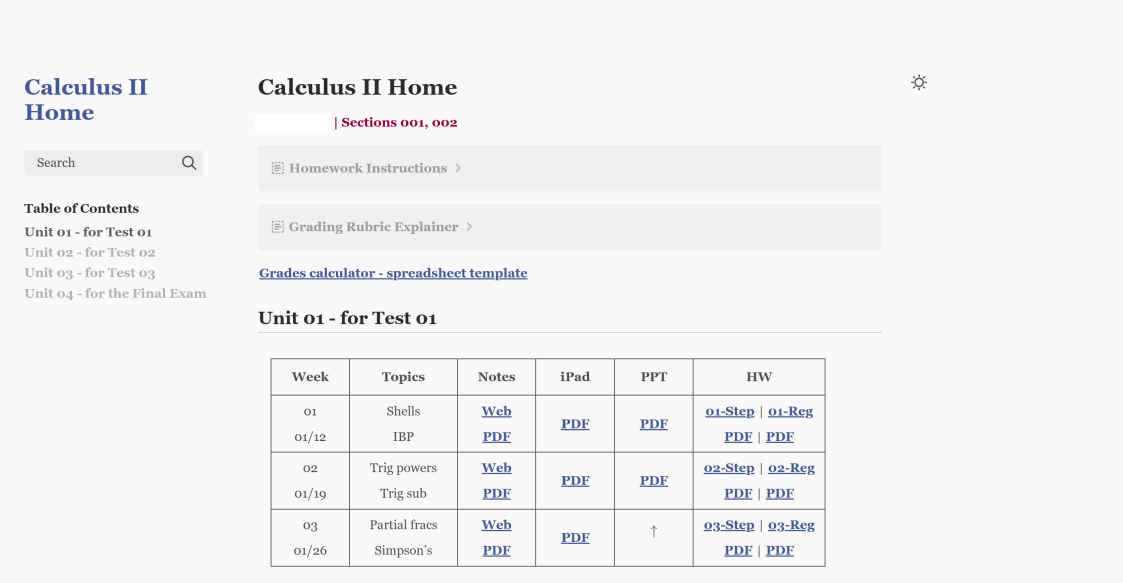

Figure 2: Course website: Calculus II home page with links to materials pages arranged by week. Tables are easy to manage in Obsidian.

Variants and alternatives

Variants on our solution within Type W3 are also reasonable. For example, instead of using Cloudflare Pages, one can use GitHub Pages for site hosting as well as for the content repository. We chose Cloudflare Pages because it does not expose all source files to the public. At present a GitHub repository is either completely public or private, and only public repositories are publishable through GitHub Pages. By keeping our GitHub repository private, we can include in the one main repository any homework solutions, exams, and other contents we do not wish to make public at all times. It is also possible to set up password protection for a site for free using a Cloudflare Worker, although the setup for this is more complex.

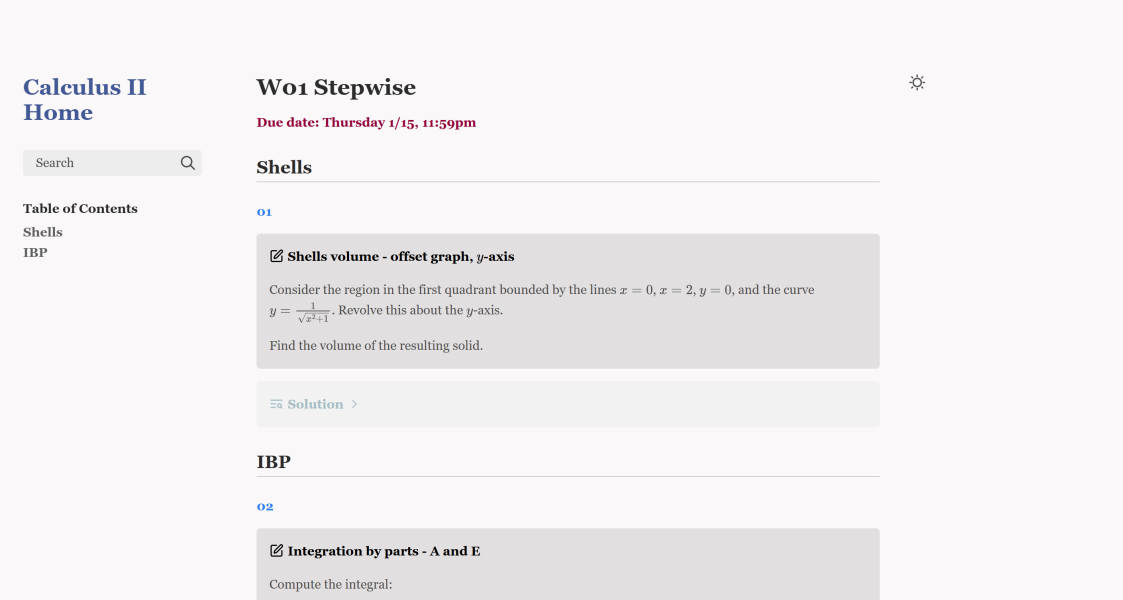

Figure 3: Course website: Sample homework page showing expandable solution call-outs. Solutions displayed in the callouts on a given homework page are easily published or hidden in a batch.

For another variant system, one could use Obsidian only to create the content, and then export each file to HTML (via an Obsidian plugin) and load the HTML files directly into an LMS such as Canvas. This would provide institutional access control through the LMS, but it would forego much of the content management functionality and the efficiency of automated publishing.

An important motivation for our system, and one reason we believe it could improve adoption of DA-compliant materials, is that it holds promise to benefit *all* students — not just those who rely on accessibility features. Ease of accessing and navigating a well-organized course website may translate to more frequent and efficient study for all students, and ease of managing content may reduce the friction experienced by instructors making continual improvements to materials. We examine evidence for these hypotheses in the next section.

Student Impact: Evidence from Calculus II

This section presents an empirical study assessing the impact on students of the system described above. We examine student impact in two dimensions: student performance (assessed through overall course grades) and student experience (assessed through a survey). We emphasize that this study did not attempt to assess the *degree* of DA compliance of the generated course materials, nor the experience of low-vision students specifically. (We consider MathML as a proxy for DA compliance of math course materials; other aspects of DA standards are not specific to math.) The goal of this empirical study was to determine whether use of the system was associated with a positive, negative, or indeterminate impact on the general student population.

Setup

Our study involved a multi-section Calculus II course over 6 semesters, "S23" (Spring 2023) through "F25" (Fall 2025), in 31 distinct sections. An average of 197 students per semester (range: 145-272) enrolled in 4-7 sections, with section sizes averaging 38 students (range: 18-48, smallest three being 18, 25, 29). All 6 instructors were full-time permanent teaching staff, either senior lecturers or teaching-track professors (none were grad students, postdocs, or adjunct).

All students in all sections took the same four exams per semester (three midterms and one cumulative final). These exams determined 70-75% of each student's grade. Exams were conducted in person without technology access. Exams were drafted by one instructor in S23-S24 and by a different instructor in F24-F25; both of these instructors had taught this course many times. Homework problems were mostly the same across sections, and included both Cengage WebAssign problems and handwritten work. Instructors differed in lecture style (at least one used a flipped classroom), lecture notes, use of class time, and experience with the course.

We divide the 31 sections into "control" and "treatment" groups. Control sections (26 total) did not in any way use the course website we described, instead they offered PDF files through Canvas for lecture slides, in-class worksheets, and solutions. Treatment sections (5 total, starting F24) were taught by a single instructor who used the system described in this article in a phased rollout that we explain now. This phased rollout is important for interpreting our data.

In F24, the treatment instructor used Obsidian to type Markdown+TeX files for course materials for two sections (i.e. for lecture notes and written homework problems), and then PDFs were generated from these source files and offered to students inside Canvas, the institutional LMS (the same delivery format used by control sections). The F24 semester therefore serves as a same-instructor baseline for the subsequent treatment semesters: the instructor used the proposed authoring tools but did not yet provide students with the content organized on webpages. Note, however, that F24 was also the treatment instructor's first semester on this particular course.

In S25, there was one treatment section (with 31-34 students) and those students received all materials on a course website produced from the Markdown+TeX source files, although homework solutions were still provided as scanned PDFs. In F25, the (two) treatment sections received all materials in a website generated by Quartz (including solutions available directly below problems, after submission deadlines, as shown in Figure 3) using the full system as described in the System Overview above.

One additional institutional change must be noted. Between S25 and F25, a policy change required students to hold formal college credit for Calculus I before enrolling in Calculus II. This change might be expected to improve the preparation of students equally in both treatment and control groups in F25. We return to this point in the discussion.

Data on student performance was obtained from the internal Office of Institutional Research & Analytics (IRA). See Table 3 in the Appendix for complete source data as used in our analysis. For each section, we gathered and computed: (i) the percentage of students receiving A-band grades (A+, A, A−), (ii) the percentage receiving DFW, and (iii) the GPA as an average of standard grade-point values, averaged over students in the section.

Data on student experience was collected through a survey administered by a central university office to the F25 cohort at the end of the semester. The survey drew 55 responses from treatment students (out of 89 in that group) and 16 from control students (also out of 87 in that group). The central office administers such surveys as part of a broad study authorized by the IRB at this university.

Performance results

In Figure 4 we depict the section GPA averaged across sections in each semester. Blue bars represent control section averages; red bars represent treatment section averages. By semester, the control bars account for 4, 6, 5, 5, 4, and 2 sections; the treatment bars account for 2, 1, and 2 sections (starting F24).

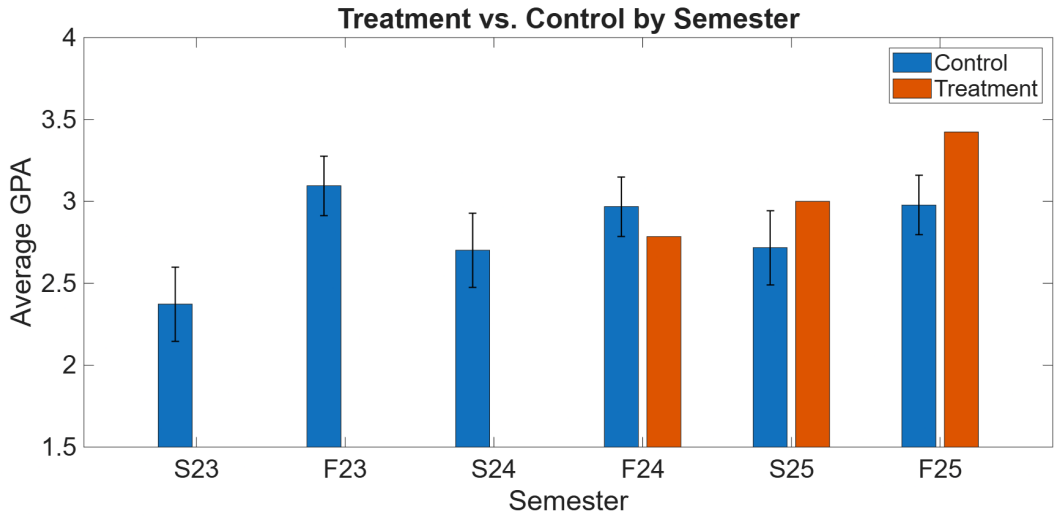

Figure 4: GPA by semester. Blue bars: control sections; red bars: treatment sections. Error bars represent σ_{fall} or σ_{spring}, the population standard deviation of control section GPAs by season. These error bars indicate the typical spread of section GPAs around the seasonal control average.

The error bars on control sections represent the population standard deviation of all control sections in the corresponding season (σ_{fall} computed over 13 fall sections, σ_{spring} over 13 spring sections). We separate seasons because we observe a stable pattern of stronger performance in fall cohorts than in spring cohorts, which is visible in the oscillating blue bars.

Three features of the data are notable:

1. The F24 treatment bar (same-instructor baseline with PDF delivery) sits approximately σ_{fall} *below* the control average, consistent with the instructor's first semester on the course.
2. The S25 treatment bar (first semester using a course website) exceeds the control average by more than σ_{spring}, and is higher than the F24 treatment bar despite the typical pattern of *lower* grades in spring semesters.
3. The F25 treatment bar (full system operational) is 2.46 increments of σ_{fall} above the F25 control bar, a significant separation.

The jump in performance between F24 and S25 is particularly informative. Since the same instructor taught both semesters using Markdown+TeX source materials, but switched from PDF delivery (F24) to website delivery (S25), the performance improvement between these semesters, and especially the reversal of the usual fall-to-spring decline, is consistent with the course website making a meaningful difference.

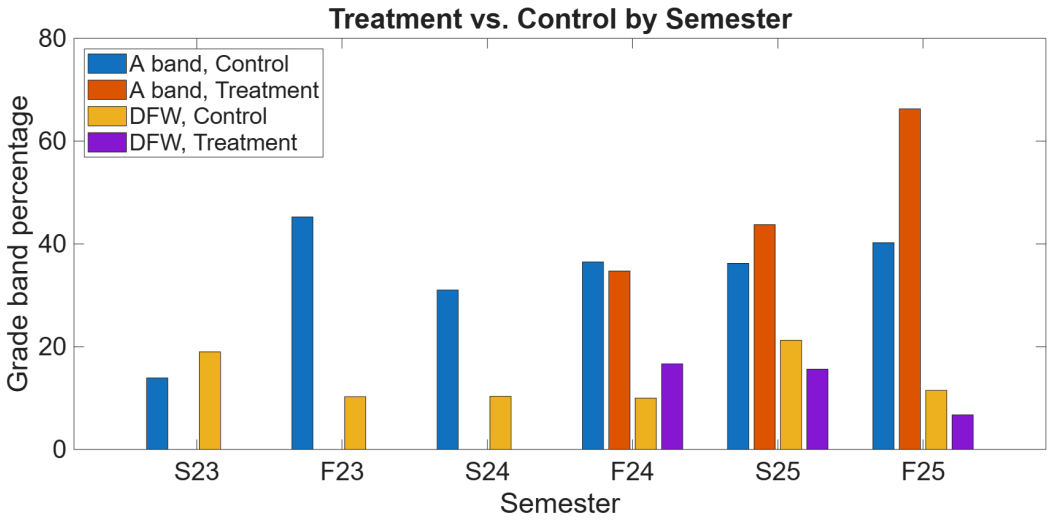

Figure 5: A-band and DFW percentages by semester.

Figure 5 depicts the percentages of students receiving A-band grades and DFW outcomes. The A-band growth in treatment sections is even more pronounced than the GPA trend. The treatment DFW rate also steadily decreases and is lower than the control in S25 and F25, when the course website was in use.

Survey results

The centrally administered survey included the following questions relevant to this study:

- Q_1 : "The primary course materials of my calculus class (notes, slides, videos, worksheets, problems, textbook) offered in my course/section were very helpful overall."
- Q_2 : "It was easy to understand the course materials when I viewed and studied them the first time through."
- Q_3 : "It was easy to navigate the course materials while I was studying later on for exams or solving problems."
- Q_4 : "Ease of navigation of course materials was important for supporting my problem-solving work."
- Q_5 : "Ease of navigation of course materials was important for supporting my review study for exams."

Respondents selected one option from the list: "Strongly agree," "Agree," "Neutral," "Disagree," "Strongly disagree." We analyzed responses using two coding schemes. In *collapsed coding*, responses were mapped to three categories reflecting direction of agreement: "Strongly agree" and "Agree" $\rightarrow$ 3, "Neutral" $\rightarrow$ 2, "Disagree" and "Strongly disagree" $\rightarrow$ 1. This coding focuses on the direction of agreement rather than its intensity, which may be more robust given the small control group size. In *full Likert coding*, responses were mapped to 5 ("Strongly agree") through 1 ("Strongly disagree") in order. Results for both codings are reported in Table 2.

	Collapsed (1–3)			Full Likert (1–5)		
Q	**Treatment**	**Control**	σ_i	**Treatment**	**Control**	σ_i
Q_1	2.75	2.75	±0.52	3.95	4.06	±0.75
Q_2	2.44	2.19	±0.73	3.51	3.31	±0.85
Q_3	2.84	2.69	±0.43	4.13	4.06	±0.72
Q_4	2.76	2.69	±0.49	3.98	4.00	±0.74
Q_5	2.78	2.81	±0.41	4.09	4.19	±0.72

Table 2: Survey results: mean values and pooled standard deviations under both coding schemes. Higher values indicate stronger agreement.

The results are also depicted in Figure 6 using the collapsed coding. For each question, the error bar represents a pooled standard deviation for that question computed as:

$$\sigma_i = \sqrt{\frac{1}{55+16}\left((55)\sigma_{T_i}^2 + (16)\sigma_{C_i}^2\right)}$$

where σ_{T_i} and σ_{C_i} are the population standard deviations of the treatment and control groups' coded responses for question Q_i.

Under the collapsed coding, the treatment trends slightly higher on Q_2 and Q_3, but all differences are well within $\frac{1}{2}\,\sigma_i$ of the control. Under the full Likert coding, the treatment lags slightly below the control on Q_1 and Q_4. A Welch's t-test (one-tailed, unequal variances) yielded $p = 0.15$ for Q_2 and $p = 0.18$ for Q_3 under the collapsed coding; all other p-values were higher. So we do not see a statistically significant difference under either coding scheme.

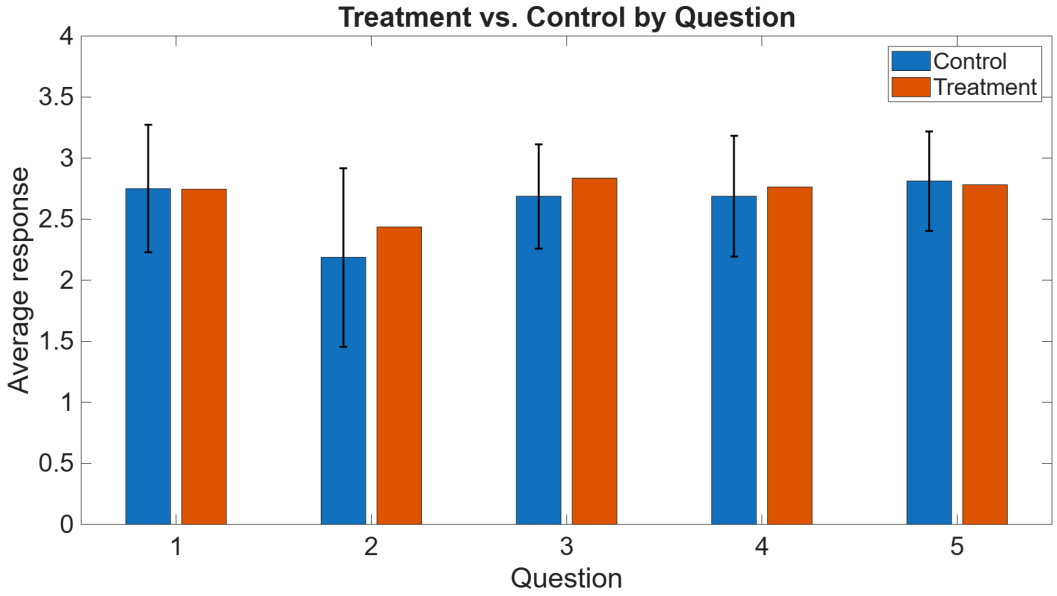

Figure 6: Survey response averages by question (collapsed coding).

Limitations

The experimental design of this study carries several important limitations. We discuss each now alongside available mitigating evidence.

Single instructor. All treatment sections were taught by a single instructor who taught no control sections, so instructor effects cannot be fully separated from treatment effects in this design. The within-instructor comparison between F24 (PDF delivery) and S25/F25 (website delivery) provides partial control for the instructor variable, since the same person taught all three semesters using very similar source materials but with different delivery methods. The performance increase between F24 and S25 coincides with the introduction of the course website rather than with a change in treatment instructor.

Instructor acclimation. The treatment instructor was new to this course in F24, so some performance improvement in the subsequent semesters may be expected as the instructor acclimated to the course content, learning objectives, and the style and expectations of the common assessments. The course materials were also adjusted in minor ways between F24 and S25 to better align with the common exams, and the use of class time shifted from blackboard lecturing (F24) to annotating projected notes from an iPad (S25, F25). To help assess whether instructor acclimation alone could account for the observed trajectory, Figure 7 shows the performance over comparable multi-semester periods for two other instructors who were also new to this course during the study window but who did not implement the system described in this article. (New instructor 1 taught this course only F23, S24, F25, while New instructor 2 taught only F24, S25.) Neither shows performance gains of comparable magnitude relative to the control, suggesting that acclimation to the course is not sufficient on its own to explain the treatment group's trajectory.

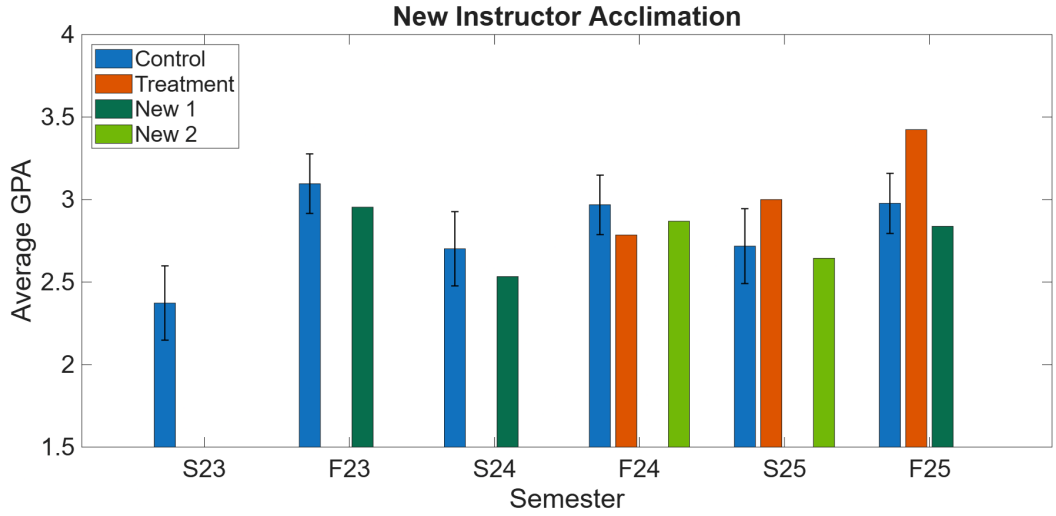

Figure 7: Semester-by-semester performance for two other instructors who were new to the course during the study period and who did not implement the proposed system, alongside Figure 4 bars. Neither shows gains comparable to the treatment group, relative to the control.

Prerequisite policy change. Between S25 and F25, the institution began requiring formal Calculus I credit for enrollment in Calculus II, which may have improved the preparation of F25 students. However, this policy change would be expected to affect both treatment and control groups equally. Inspecting the control bars in Figure 4, we observe that the control group GPA does not increase from F24 to F25; indeed, control GPAs are flat or slightly declining over the F23–F24–F25 sequence. This suggests that the prerequisite policy change is not the primary driver of the treatment group's performance gains in F25.

Small survey control group. The student experience survey drew only 16 responses from the control group, which significantly limits the statistical power of the survey analysis. Consequently, the null result on the survey should not be interpreted as strong evidence that the system has no effect on student experience, but neither does it indicate a negative effect. Extending the survey to additional cohorts is a goal of future work.

Discussion

Taken together, the performance data present a consistent picture. The treatment instructor's trajectory shows a marked improvement that begins precisely when the course website is introduced (S25) rather than when the instructor begins teaching the course (F24). This improvement grows further in F25 when the full system is operational including the website access to homework solutions after submission deadlines. The $2.46\,\sigma_{\text{fall}}$ separation between treatment and control in F25 is notable given that σ_{fall} reflects the normal range of variation across 6 instructors (13 fall control sections) with varying levels of course experience. A result of this magnitude would be difficult to achieve through instructor individuality or acclimation alone. Two other new instructors on the same course do not show similar gains over comparable time frames, and the prerequisite policy change does not appear to have benefited control sections. While the single-instructor design means we cannot definitively attribute the performance gains to the courseware system, the combination of these lines of evidence supports the conclusion that the system is a meaningful contributor.

The survey results do not show a statistically significant difference between treatment and control groups on any question under either coding scheme. The small control group (16 respondents) limits what may be concluded with any confidence. The data does, however, indicate that the system is unlikely to have a substantial negative effect on student experience. This is a useful finding for the adoption problem, since a negative effect on student experience would weigh against wide adoption of the system.

Future Work: Study of Adoption Barriers

Having established a DA-compliant solution and studied its effects on students, we turn to the next major question: what are the barriers to adoption of this solution by other faculty?

We propose the identification of a small group of instructors at other institutions with varied teaching contexts who are willing to adopt the system for their own courses. We will collect data through instructor surveys (before, during, and after adoption) and through records of issues encountered during the setup and maintenance process. The initial survey will assess participants'

backgrounds in programming, web development, LaTeX use, and course material creation; the final survey will assess satisfaction after several semesters using our system.

As a secondary objective, participating faculty would contribute non-private student performance data from before and after adopting the system. Aggregating this data across multiple institutions and instructors would strengthen the evidence for the student performance findings reported here by addressing the single-instructor limitation of the current study, and it would provide further motivation for broad adoption.

Acknowledgments

We thank Meiqin Li for helpful comments and discussion, and Lindsay Wheeler for helping us source grades data through the UVA IRA. We also thank the UVA Center for Teaching Excellence for supporting work by the second author.

References

[1] Fact Sheet: New Rule on the Accessibility of Web Content and Mobile Apps Provided by State and Local Governments. URL https://www.ada.gov/resources/2024-03-08-web-rule/.

[2] Web Content Accessibility Guidelines (WCAG) 2.1. URL https://www.w3.org/TR/WCAG21/#non-text-content.

[3] Mathematical Markup Language (MathML) Version 4.0, . URL https://www.w3.org/TR/mathml4/#acc_benefits.

[4] Braille Accessibility in State Assessments: Roles, Responsibilities, and Strategies. URL https://ncademi.org/provide/formats/braille-assessments/#c-use-of-mathml-for-interoperability-with-a-variety-of-assistive-technology.

[5] Accessing Math Content with JAWS and Fusion – Freedom Scientific, . URL https://www.freedomscientific.com/training/teachers/accessing-math-content-with-jaws-and-fusion/.

[6] Preston Lewis, Steve Noble, and Neil Soiffer. Using accessible math textbooks with students who have learning disabilities. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 139–146. ACM. ISBN 978-1-60558-881-0. doi: 10.1145/1878803.1878829.

[7] Erin Scanlon, Zachary W. Taylor, John Raible, Jacob Bates, and Jacquelyn J. Chini. Physics webpages create barriers to participation for people with disabilities: Five common web accessibility errors and possible solutions. 8(1):1–16. ISSN 2196-7822. doi: 10.1186/s40594-021-00282-3.

[8] Morgan Haley McKie and Alexandra Coso Strong. Mapping the landscape of digital accessibility in computer science education: A mapping literature review. In *2024 ASEE Annual Conference & Exposition*. ASEE Conferences. doi: 10.18260/1-247760.

[9] Jason J. G. White. The Accessibility of Mathematical Notation on the Web and Beyond. 23(1):1–14. ISSN 1940-9923. doi: 10.14448/jsesd.12.0013.

[10] MathML — MDN, . URL https://developer.mozilla.org/en-US/docs/Web/MathML.

[11] Micha l Maćkowski, Piotr Brzoza, Marek Żabka, and Dominik Spinczyk. Multimedia platform for mathematics' interactive learning accessible to blind people. 77(5):6191–6208. ISSN 1380-7501. doi: 10.1007/s11042-017-4526-z.

[12] Amjad Ali, Shah Khusro, and Tahani Jaser Alahmadi. Accessible interactive learning of mathematical expressions for school students with visual disabilities. 10. ISSN 2376-5992. doi: 10.7717/peerj-cs.2599.

[13] Neil Soiffer. MathPlayer v2.1: Web-based math accessibility. In *ASSETS '07: Proceedings of the Ninth International ACM SIGACCESS Conference on Computers and Accessibility: Tempe, Arizona, USA, October 15-17, 2007*, pages 257–258. ACM. ISBN 978-1-59593-573-1. doi: 10.1145/1296843.1296900.

[14] Harvey J. Hindin. MATHTYPE 6.6. 44(3):273. ISSN 0730-8639.

[15] Donal Fitzpatrick, Azadeh Nazemi, and Grzegorz Terlikowski. EuroMath: A Web-Based Platform for Teaching of Accessible Mathematics. In *Computers Helping People with Special Needs*, volume 12376 of *Lecture Notes in Computer Science*, pages 385–392. Springer International Publishing AG. ISBN 978-3-030-58795-6. doi: 10.1007/978-3-030-58796-3_45.

[16] Changxu Duan. Layout-Aware Text Editing for Efficient Transformation of Academic PDFs to Markdown. In *Document Analysis and Recognition – ICDAR 2025*, Lecture Notes in Computer Science, pages 221–241. Springer Nature Switzerland. ISBN 978-3-032-04613-0. doi: 10.1007/978-3-032-04614-7_13.

[17] Frank Mittelbach, Ulrike Fischer, David Carlisle, and Joseph Wright. MathML and other XML Technologies for Accessible PDF from LATEX. In *Proceedings of the 2025 ACM Symposium on Document Engineering*, pages 1–4. ACM. ISBN 979-8-4007-1351-4. doi: 10.1145/3704268.3748669. URL https://www.latex-project.org/publications/indexbyyear/2025/.

[18] Accessible math in PDF – finally! – PDF Association, . URL https://pdfa.org/accessible-math-in-pdf-finally/.

[19] The LaTeX Tagged PDF Project, . URL https://latex3.github.io/tagging-project/.

[20] LaTeX package/class tagging status, . URL https://latex3.github.io/tagging-project/tagging-status/.

Appendix

Term	Sec.	A+	A	A–	B+	B	B–	C+	C	C–	DFW	N
2023 Spring	1	1	2	3	2	0	1	3	2	7	11	32
	2	1	2	0	6	7	3	6	4	6	8	43
	3	1	4	3	7	8	5	1	5	2	7	43
	4	1	1	3	6	3	4	3	5	10	4	40
2023 Fall	1	3	7	10	3	7	3	2	4	2	3	44
	2	2	8	7	6	6	4	2	3	1	4	43
	3	8	9	7	3	5	4	1	4	0	4	45
	4	6	15	7	5	6	1	0	0	2	4	46
	5	2	9	6	4	7	2	1	1	0	8	40
	6	1	7	5	8	10	3	1	5	1	4	45
2024 Spring	1	1	7	4	5	2	3	6	6	0	1	35
	2	1	5	5	2	3	4	4	1	4	3	32
	3	3	2	3	2	3	1	3	4	2	6	29
	4	2	1	4	5	1	3	4	4	5	2	31
	5	1	2	4	0	0	1	1	2	4	3	18
2024 Fall	1	2	9	1	5	9	4	1	4	4	5	44
	2	3	15	0	5	6	6	0	5	1	1	42
	3	6	11	5	5	5	2	1	2	2	4	43
	4	3	4	3	6	4	6	3	4	3	6	42
	5	1	4	9	2	4	3	1	5	2	7	38
	6	2	2	7	4	6	4	2	2	0	5	34
	7	1	9	1	4	3	1	2	2	2	4	29
2025 Spring	1	3	7	4	5	3	1	1	2	1	5	32
	2	5	2	5	2	3	5	1	5	1	7	36
	3	3	6	3	4	5	1	3	2	2	5	34
	4	5	3	6	1	1	1	3	3	1	8	32
	5	2	2	4	2	4	2	1	1	0	7	25
2025 Fall	1	7	18	8	3	1	2	0	0	2	3	44
	2	3	13	10	6	5	3	0	0	2	3	45
	3	3	8	9	5	5	4	1	2	0	5	42
	4	0	7	8	2	10	4	3	2	4	5	45

Table 3: Complete grade distributions by term and course section gathered from the UVA IRA (ira.virginia.edu/university-data-home). **N** denotes section enrollment. DFW combines D, F, and withdrawal outcomes. Treatment sections in 2024 Fall are 5, 6; in 2025 Spring just Sec. 1, and in 2025 Fall they are 1, 2.