LoadTrace: An Artificial Intelligence Web Application for Visualizing Gravity Load Takedown in Structural Engineering Education

Derrick Mirindi[1,2]*, David Sinkhonde[3], and Frederic Mirindi[4]

[1] PhD Candidate, 1700 East Cold Spring Lane, Baltimore, Maryland 21251-0001, USA
[2] Graduate Researcher, University of Pennsylvania, Philadelphia, PA 19104, USA
[3] Graduate Researcher, Pan Africa University Institute for Basic Sciences, Technology and Innovation, P.O. Box 62000 00200 Nairobi, Kenya
[4] PhD Candidate, 400 Dysart Road, Winnipeg, Manitoba, R3T 2N2, Canada

Abstract

We present LoadTrace, a browser-based educational web application designed to improve how students learn gravity load takedown in multi-story building structures. The platform combines a parametric structural model, real-time visualization, stepwise explanation, and an artificial intelligence (AI)-supported tutoring agent in a single interface publicly deployed at https://derrickmirindi.github.io/loadtrace/. Rather than treating structural load analysis as a static sequence of hand calculations, the application reconfigures it as an interactive and interpretable learning process in which each floor's contribution to the total building load can be traced visually and numerically. We frame LoadTrace as a design-science artifact for structural engineering education rather than as a professional design package or a code-compliance engine. Its contribution lies in integrating three layers that are often separated in existing teaching tools: a transparent computational core, a narrative visual interface, and an AI-based conversational explainer grounded in the current model state. The application uses HTML (HyperText Markup Language), CSS (Cascading Style Sheet), JavaScript, and SVG (Scalable Vector Graphic) for the front-end implementation, while the conversational agent relies on Gemini-style API (Application Programming Interface) integration practices consistent with current developer guidance on API key handling and restriction. The visual structure of the interface also draws from grid-based editorial design principles informed by the Hyperagent public skill resource `skill-muller-brockmann-grid-systems.json`, which was used as a design reference for layout discipline and typographic alignment. In addition, we argue that the significance of LoadTrace is not merely computational. Its broader value lies in demonstrating how AI evolution can reshape structural engineering education by making canonical load-path concepts more explorable, interpretable, and dialogic. In this sense, the application proposes a model for the next generation of structural learning environments: tools that do not replace engineering reasoning but amplify it through visual intelligence, interaction, and guided explanation.

Keywords: structural engineering education, gravity load takedown, AI tutoring, educational web application, load visualization.

1 Introduction

Gravity load takedown is one of the foundational analytical procedures in structural engineering because it connects the notion of design loads to the actual physical path by which forces move

*Corresponding author.

from slabs and roofs into beams, columns, walls, and foundations. Yet students frequently learn this topic through static worksheets, isolated equations, and simplified tables that fail to convey the cumulative and spatial nature of the process. As a result, many can replicate arithmetic steps without developing a durable intuition for how a building "collects" and transfers its own weight and imposed loads through the structure.

This pedagogical problem has become more visible in an era where engineering students increasingly expect software-mediated feedback, visual interfaces, and conversational support rather than purely sequential instruction. Recent scholarship on structural engineering education has shown that interactive learning systems and immersive tools can improve conceptual access, but many published systems remain narrow in scope, low in interactivity, or insufficiently integrated with explanatory feedback [1, 2]. At the same time, work in design science research has established that educational software artifacts can be studied rigorously when they are positioned as purpose-built solutions to a defined instructional problem [3, 4].

LoadTrace was developed in response to this intersection of need and opportunity. The project asks a simple but consequential question: what happens when gravity load takedown is redesigned not as a calculation sheet, but as a visual, interactive, and AI-assisted learning environment? The resulting application enables users to define a simplified building, assign floor and roof loads, choose ASD (Allowable Stress Design) or LRFD (Load and Resistance Factor Design) combinations, and then observe the cumulative gravity load increase story-by-story down to the foundation. At the same time, the application supports local interpretation through tables, dynamic highlighting, and short guided explanations, while also embedding an AI agent capable of responding to natural-language questions about the current structural model.

The central claim of this work is that such a tool can contribute meaningfully to structural engineering education because it transforms a procedural exercise into a hybrid cognitive environment. The student is no longer only solving for reactions; the student is navigating a computational representation of the building, observing a visual load pathway, and querying an explanatory agent about the engineering meaning of the result. This shift is important because AI is often discussed in structural engineering in terms of optimization, prediction, or automation, whereas its educational potential may be equally transformative when used to enrich conceptual understanding [5, 6].

Accordingly, this study presents LoadTrace as an educational artifact and analyzes its conceptual design, implementation logic, and scholarly significance. It does not claim that the tool replaces detailed structural design software or formal code checks. Instead, it argues that AI-enhanced web applications such as LoadTrace can reshape the teaching of structural building design by making core analytical ideas more legible, more interactive, and more interpretable for learners.

2 Background and Conceptual Framing

The educational challenge addressed by LoadTrace is not simply that gravity load calculations are difficult. Rather, the challenge is that the dominant mode of instruction often separates numerical operations from the spatial and architectural reality they represent. Students may know that floor loads should be added from top to bottom, yet still fail to internalize why the cumulative demand at lower levels must increase, how roof-specific loading affects downstream force transmission, or why load combinations alter perceived structural demand. In that sense, the issue is epistemic as much as procedural: the learner lacks a convincing representation of force flow.

This challenge aligns closely with studies on visualization and interactive learning in engineering education. A recent study on augmented reality for structural analysis emphasized the importance of improving conceptual understanding through interactive visual tools [1]. Other educational prototypes in structural engineering visualization have shown that students benefit when spatial and structural concepts are staged through progressive interaction rather than presented as final results alone [2]. These findings support the premise that the most effective learning tools are not merely illustrative; they are participatory.

The methodological framing of the present work is grounded in design science research. The design science research (DSR) treats artifacts as legitimate knowledge contributions when they are intentionally designed to solve identified problems and when their construction and rationale are made explicit [3, 4]. This is particularly relevant for educational software because many useful innovations in teaching do not originate as abstract theories; they emerge as practical artifacts that embody pedagogical assumptions, interaction models, and domain logic. LoadTrace is thus best understood as a design-science artifact that integrates engineering content, visual communication, and AI-mediated explanation into a single research object.

The AI dimension adds a further conceptual layer. Structural engineering scholarship increasingly recognizes AI as major forces in analysis, monitoring, and optimization [6]. Yet educational applications remain comparatively underexplored. When AI is used not to generate final designs but to contextualize and explain deterministic calculations, it can function as a cognitive bridge between formal engineering logic and student understanding. LoadTrace adopts this orientation deliberately: the structural results are produced by transparent formulas coded directly in JavaScript, while the AI component operates as a conversational explanatory layer rather than as an opaque analytical engine [5, 7].

3 Research Contribution and Innovation

The innovation of LoadTrace lies in how it reconceives a conventional engineering topic as a multimodal learning system. Existing instructional practice often separates input definition, numerical calculation, conceptual explanation, and interface presentation into distinct media such as lecture notes, spreadsheets, slides, and office-hour discussion. LoadTrace collapses these fragments into a single computational environment where structural inputs, calculations, visual output, and explanatory dialogue coexist in real time. This integration is not merely convenient; it changes the epistemic structure of the learning experience.

Three contributions are especially important. First, the application transforms load takedown into an explicitly traceable process. Rather than giving only the final base reaction, it reveals how each story contributes incrementally to the cumulative load below. Second, it introduces an interpretable AI tutoring layer that can answer context-specific questions using the active model state, allowing students to move fluidly between structured input and open-ended inquiry. Third, it demonstrates that high-quality educational engineering interfaces can be created with web-native technologies and disciplined visual systems, without dependence on proprietary simulation platforms or immersive hardware [1, 3, 8].

This design philosophy also makes the tool adaptable. Because the application is deployed through a standard browser, it can be embedded in teaching modules, used in flipped classrooms, demonstrated in lectures, or assigned as a self-study companion. Its interface is clear enough for beginners, yet transparent enough for advanced learners to inspect its assumptions and critique its simplifications. In that sense, the application does not merely teach one topic; it also models a broader approach to engineering pedagogy in which analytical concepts are presented through interactive and explainable computational artifacts.

4 System Overview

LoadTrace is publicly available as a browser-based application at https://derrickmirindi.github.io/loadtrace/ [9]. It is implemented as a single-page web application and organized around three coordinated regions: a parameter-entry zone, a structural visualization and explanation zone, and a conversational agent zone. This arrangement reflects the pedagogical sequence of the tool itself: define the building, compute the takedown, inspect the evolving structure, and then ask interpretive questions.

The input panel allows the user to specify the number of stories, floor area, dead load, live load, roof dead load, snow load, facade load, story height, perimeter, number of columns, and load-combination type. These values populate a JavaScript data structure that serves as the live model state for the app. The middle panel displays the story-by-story building diagram in SVG, alongside narrative text and a numerical table. The right panel hosts the LoadTrace Agent, which accepts natural-language questions about the current model and returns short explanatory answers.

The overall architecture is intentionally transparent. Instead of concealing the computational process behind black-box outputs, the application exposes the story-level and cumulative quantities directly. This transparency is vital in an educational setting because the objective is not only to produce answers, but also to make the formation of those answers visible, inspectable, and discussable.

5 Structural Model and Load Formulation

The computational logic of LoadTrace is based on a simplified multi-story gravity load model designed for conceptual learning rather than detailed design. Let the building have n stories, floor plan area A, typical floor dead load D, typical floor live load L, roof dead load D_r, snow load S, facade surface load intensity f, story height h, building perimeter P, and N_c columns. These quantities are provided by the user through the interface and converted into story-level and cumulative structural actions.

To keep the model direct and interpretable, each story is represented by a small set of load components. The facade contribution per story is defined as

$$F_{\text{fac}} = fPh. \tag{1}$$

For instructional simplicity, the app treats interior floors and the roof separately. Each story is assigned total dead, live, and snow components, and then a factored story load is computed according to the selected design method. For the selected combination coefficients α, β, and γ, the story-level factored load is

$$W_i = \alpha D_i + \beta L_i + \gamma S_i. \tag{2}$$

The key pedagogical quantity is the cumulative gravity load carried at level ℓ, obtained by summing all story loads from the roof down to that level:

$$W_{\text{cum}}(\ell) = \sum_{i=\ell}^{n} W_i. \tag{3}$$

This cumulative expression captures the central concept the application is designed to teach: lower levels attract larger vertical force because they must support all tributary stories above

them. Once the cumulative level load is known, the average load per column is computed as

$$w_c(\ell) = \frac{W_{\text{cum}}(\ell)}{N_c}. \tag{4}$$

At the foundation, the total gravity load is therefore

$$W_{\text{base}} = \sum_{i=1}^{n} W_i, \tag{5}$$

with the corresponding average base load per column

$$w_{c,\text{base}} = \frac{W_{\text{base}}}{N_c}. \tag{6}$$

The model also reports an equivalent supported mass

$$m = \frac{W_{\text{base}}}{g}, \tag{7}$$

where $g = 9.81\,\text{m/s}^2$. This mass conversion is not used for design calculations, but it helps students develop intuition for the magnitude of the total force reaching the foundation.

Two load-combination options are implemented in the interface: a simplified ASD form and a simplified LRFD form. These combinations are intentionally reduced versions of broader code logic so that students can compare service-level and factored-level responses without being overwhelmed by combinatorial complexity. This simplification is consistent with the educational purpose of the application, which is to support conceptual understanding of load accumulation rather than to produce construction-ready designs.

6 Implementation Details

6.1 Front-end Structure and Styling

The front end is implemented as a single HTML document with embedded CSS and JavaScript, a choice that reinforces portability, inspectability, and ease of classroom deployment. The interface is organized using CSS Grid into a three-column layout: the left column contains the input form, the center column contains the SVG-based structural visualization and narrative explanation, and the right column contains the AI agent. This arrangement was designed to support a natural learning flow from parameter definition to structural interpretation to reflective questioning.

The webpage design adopts a restrained editorial visual language based on Inter and Space Mono, a strong baseline rhythm, and a visible modular grid. The layout logic was informed by the Hyperagent public skill `skill-muller-brockmann-grid-systems.json`, which served as a design reference for disciplined column structure, typographic alignment, and visual hierarchy in the interface [8]. The result is a front end that does more than display information; it stages engineering reasoning through a graphic system that communicates order, sequence, and cumulative structure.

If the interface uses C columns, gutter width g, margin M, and usable width W_u, then the effective column width can be approximated as

$$w_{\text{col}} = \frac{W_u - (C-1)g}{C}. \tag{8}$$

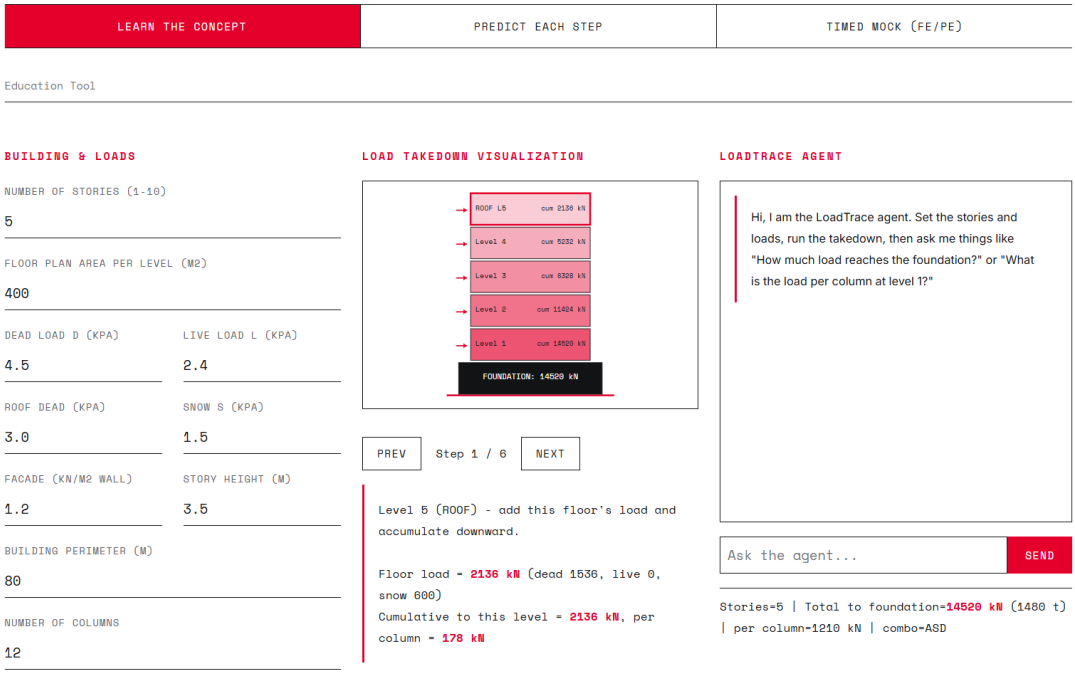

Figure 1: Front-end structure of the LoadTrace interface, showing the single-page HTML architecture, three-column instructional layout, SVG load visualization workflow, and AI-supported agent region.

Likewise, if the baseline increment is b, then vertical rhythm can be represented as a repeating sequence

$$y_k = kb, \quad k \in \mathbb{N}. \tag{9}$$

These geometric relations are not structural formulas in the engineering sense, but they are relevant to the pedagogical performance of the application because they support clarity, consistency, and cognitive legibility across the interface.

6.2 JavaScript Model and Computational Flow

The computational core is implemented in JavaScript and centered on a `run()` function that reads user inputs, constrains the number of stories to an admissible range, computes story-level and cumulative quantities, and updates the visualization, table, and explanation panels. Input parsing is handled through helper functions that convert form values into numeric quantities. If a user-provided value x is parsed from the DOM, its effective numerical value can be written as

$$\hat{x} = \max(0, \text{parseFloat}(x)). \tag{10}$$

The number of stories is then clamped to the interval $[n_{\min}, n_{\max}]$:

$$n = \min\left(n_{\max}, \max(n_{\min}, \hat{n})\right). \tag{11}$$

Within the model object, a `levels` array stores per-story data records. This structure is significant because it functions as a unifying data contract across the application. The same array powers the numerical table, the step explanations, the SVG labels, and the context sent to the conversational agent. In other words, the app's educational coherence is achieved partly through a deliberately shared computational representation.

The roof-to-base update loop can be interpreted algorithmically as an iterative accumulation process. If $W_{\mathrm{cum}}(\ell+1)$ is known, then the cumulative load at the next lower level is

$$W_{\mathrm{cum}}(\ell) = W_{\mathrm{cum}}(\ell+1) + W_{\ell}. \tag{12}$$

This recursive form expresses the same logic as the summation formulation, but it also reflects how the application actually computes the load takedown in code. It is especially useful pedagogically because it highlights the local-to-global structure of the process: each story contributes a new increment to what is already being carried below.

6.3 SVG Visualization and Visual Encoding

The building is rendered in an inline SVG canvas as a sequence of stacked rectangles, one for each story plus a foundation band. If the total drawable height is H, then the vertical band height is approximated by

$$h_b = \frac{H}{n+1}. \tag{13}$$

If y_0 is the top offset, the vertical position of the i-th story band is

$$y_i = y_0 + (i-1)h_b. \tag{14}$$

The color intensity of each story is normalized by the total foundation load so that the visual weight of a level is proportional to its cumulative carried force. This ratio is defined as

$$r_{\ell} = \frac{W_{\mathrm{cum}}(\ell)}{W_{\mathrm{base}}}. \tag{15}$$

To convert this engineering ratio into a display opacity, the interface applies a linear intensity mapping of the form

$$o_{\ell} = o_{\min} + (o_{\max} - o_{\min})r_{\ell}. \tag{16}$$

This mapping is pedagogically effective because it allows the graphic representation to communicate an engineering truth immediately: lower stories appear darker because they carry more cumulative load. The visualization thus functions not as decorative output, but as a semiotic instrument that compresses a table of values into an intuitive vertical gradient.

The highlighted step indicator uses a simple conditional display logic. If the current step corresponds to level ℓ, the stroke width of that story rectangle becomes

$$s_{\ell} = \begin{cases} s_{\mathrm{active}}, & \text{if level } \ell \text{ is selected,} \\ s_{\mathrm{base}}, & \text{otherwise.} \end{cases} \tag{17}$$

This coupling of narrative step and visual emphasis is central to the learning design because it synchronizes explanation and geometry. The student does not merely read that a certain level carries a certain load; the student sees the corresponding region of the structure become active in the drawing.

6.4 Gemini API Key and Agent Integration

The LoadTrace Agent extends the application from a numerical visualizer into a conversational educational environment. The agent accepts short natural-language questions about the current structural model and returns concise engineering explanations grounded in the live application state. In implementation terms, the user enters a Gemini API key through the interface, and the system forwards a prompt containing both a system instruction and the current model context to an external AI endpoint. Google's developer documentation states that Gemini API keys can be created through Google AI Studio and that developers should avoid exposing keys directly in client-side production workflows [7]. Google Cloud guidance further emphasizes restricting API access, protecting keys, and storing sensitive credentials in secure systems rather than where they can be easily seen [10].

The interaction can be represented conceptually as a two-stage pipeline. First, the structural engine computes the model state from the user inputs:

$$\mathcal{M} = \mathcal{F}(\mathbf{p}), \tag{18}$$

where $\mathbf{p}$ denotes the vector of building and loading parameters. Second, the AI layer generates a context-aware explanation from the user query q and model state $\mathcal{M}$:

$$e = \mathcal{A}(q, \mathcal{M}). \tag{19}$$

This separation is crucial to the scholarly framing of the tool. The engineering calculations remain deterministic, explicit, and inspectable, while the AI component serves as an interpretive layer that translates computed outputs into concise pedagogical language. In this sense, the agent does not replace structural reasoning; it augments the accessibility of structural reasoning.

7 Discussion and Limitations

The broader significance of LoadTrace lies in how it reframes engineering education as an interaction between formal calculation, visual representation, and machine-mediated explanation. The application suggests that the future of structural learning tools may not depend primarily on greater analytical complexity, but on more intelligent coordination between transparent models and responsive interfaces. This matters because the gap between "knowing the formula" and "understanding the force path" is often where structural learning fails.

The project also offers a more nuanced account of AI in engineering than narratives of wholesale automation. In LoadTrace, AI is not the author of the structural model. It does not invent loads, replace combinations, or size members. Instead, it plays a narrower but pedagogically powerful role: it helps students ask better questions of an already explicit model. That distinction is important for engineering education because it preserves disciplinary rigor while still leveraging the accessibility gains of conversational systems [5, 6, 7].

Another contribution of the project is methodological. By combining design-science reasoning with careful front-end composition, the application shows that interface design is not peripheral to engineering education research. The use of a modular Swiss-style grid, clear typographic hierarchy, and coordinated visual encoding demonstrates that pedagogical effectiveness depends partly on how analytical logic is staged in the interface, not merely on whether the formulas are correct. In that respect, LoadTrace advances a richer conception of structural education technology—one in which design, computation, and explanation are inseparable.

Several limitations remain. The current structural model assumes regular geometry, uniform floor areas, simplified combinations, and equal average per-column load sharing. These

assumptions are appropriate for conceptual teaching but insufficient for full design practice. The agent also depends on an external AI service, and its responses require careful control because conversational systems may provide incomplete or overconfident explanations if not properly grounded in the computed model. Most importantly, the application has not yet been deployed to students in a formal classroom or laboratory setting. Future work should deploy LoadTrace with structural engineering students and evaluate its impact through pre- and post-concept inventories, usability surveys, task-completion studies, and instructor review. Such student-centered testing is necessary to determine whether the application improves conceptual understanding, reduces common misconceptions about cumulative load paths, and supports meaningful learning beyond the demonstration level.

8 Conclusion

LoadTrace demonstrates that a structural engineering topic as classical as gravity load takedown can be reimagined through contemporary web technologies and AI-assisted explanation without sacrificing analytical transparency. The application combines a deterministic computational core, a disciplined visual interface, and a conversational tutoring layer to make load accumulation more legible, more interactive, and more conceptually durable for students [1, 3]. By embedding the logic of structural force flow into both the numerical model and the visual composition of the interface, the platform turns an abstract sequence of calculations into an exploratory learning environment.

More broadly, the project advances an educational argument: the next generation of structural teaching tools should not aim only to compute faster, but to explain better. When engineering models are paired with careful visual design and context-aware AI interpretation, students gain access not only to answers but to meaning. In that sense, LoadTrace offers a concrete example of how AI evolution can reshape structural building design education—not by replacing the engineer, but by enriching how the engineer learns to see, trace, and question the behavior of buildings.

References

[1] Miner, N., Karabulut-Ilgu, A., Jahren, C., & Alipour, A. (2025). Advancing civil engineering education: Implications of using augmented reality in teaching structural analysis. *Computer Applications in Engineering Education, 33*(4), e70059. https://doi.org/10.1002/cae.70059

[2] Yehia, A. S., Harris, D., & Bairaktarova, D. (2024). Developing augmented reality applications to help engineering students learn spatial structural engineering concepts. *American Society for Engineering Education (ASEE). Annual Conference and Exposition.*

[3] Apiola, M., & Sutinen, E. (2021). Design science research for learning software engineering and computational thinking: Four cases. *Computer Applications in Engineering Education, 29*(1), 83–101. https://doi.org/10.1002/cae.22291

[4] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

[5] Anwar, N. (2024, August 12). *Artificial intelligence in structural engineering* [Video]. YouTube. https://www.youtube.com/watch?v=9yalsKk839U

[6] Thai, H.-T. (2022). Machine learning for structural engineering: A state-of-the-art review. *Structures, 38*, 448–491. https://doi.org/10.1016/j.istruc.2022.02.003

[7] Google AI for Developers. (n.d.). *Using Gemini API keys*. Google. https://ai.google.dev/gemini-api/docs/api-key

[8] McDonnell, A. (n.d.). *Hyperagent public skills: skill-muller-brockmann-grid-systems.json* [Computer software repository]. GitHub. https://github.com/alexmcdonnell-airtable/hyperagent-public-skills/blob/main/skill-muller-brockmann-grid-systems.json

[9] Mirindi, D., & Sinkhonde, D. (2026). *LoadTrace: Gravity load takedown* [Web application]. GitHub Pages. https://derrickmirindi.github.io/loadtrace/

[10] Google Cloud. (2026, May 20). *Securing your Gemini and Google API keys*. Google Cloud Blog. https://cloud.google.com/blog/topics/developers-practitioners/api-keys-are-open-secrets