International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Implementation of VisualizeIT: An Interactive Simulation Platform for Engineering Education

Author Details:

Kushal Shankhapal[1], Yash Suryawanshi[2], Sumit Sonawane[3], Shubham Palde[4], Pushpak Nikam[5], Dhiraj A. Birari[6]

[1–6] Department of Information Technology, MVP's K.B.T College of Engineering, Nashik, India

Corresponding Author Email: suryawanshi.yash.work@gmail.com

https://doi.org/10.55041/ijsmt.v2i4.177

Cite this Article: Shankhapal, K., Suryawanshi, Y., Sonawane, S., Palde, S., Nikam, P. & Birari, D. A. (2026). Implementation of VisualizeIT: An Interactive Simulation Platform for Engineering Education. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.177

Abstract—

Engineering students studying subjects like data structures, operating systems, and database management frequently encounter a gap between what static diagrams communicate and what they actually need to understand. Concepts involving step-by-step execution and time-dependent behavior are difficult to internalize through lecture-based instruction alone, and most available simulation tools are either too generic to align with any particular curriculum or too passive to support meaningful interaction. This paper presents VisualizeIT, a web-based learning platform offering custom interactive simulations built specifically around core engineering subjects, allowing students to manipulate inputs, step through execution, and observe system behavior directly. The platform integrates a curriculum-aware simulation library that surfaces relevant modules based on a student's current academic context, structured diagnostic quizzes with persistent scoring, and a personal progress dashboard tracking cleared modules, quiz performance, and recent activity. Core simulations spanning algorithm visualization, data structures, operating systems, and database concepts were developed and evaluated through functional testing across all major user flows. All core modules performed correctly and stably across desktop and mobile environments. VisualizeIT demonstrates that simulation-based learning in engineering education can be delivered in a curriculum-aligned, assessment-integrated form through a single accessible web-based platform, and establishes the foundation for more adaptive tools ahead.

Keywords— Algorithm visualization; Engineering education; Interactive simulations; Simulation-based learning; Web-based learning

I. INTRODUCTION

There is a particular kind of confusion that engineering students experience with certain subjects, one that does not resolve itself through re-reading. A student can follow a worked example, reproduce the answer correctly, and still have no intuitive sense of why the underlying process behaves the way it does when the inputs change. This happens most often with topics that involve dynamic, step-by-step execution, where the textbook captures the result but not the process.

The process is where understanding actually lives. Research has consistently shown that subjects involving sequential, time-dependent behavior are difficult to teach through static materials alone [1, 2], and interactive simulation has emerged as a natural response to that challenge. The more relevant question for students is whether the simulations available to them are actually built around the subjects they are studying, or whether they are general-purpose tools that happen to cover some of the same ground.

VisualizeIT was developed in response to this question, for engineering students who need to see topics in their subjects move before they can fully understand them. The platform provides custom interactive simulations built specifically around core engineering subjects, where students can manipulate inputs, step through execution at their own pace, and observe how system behavior changes in response to their decisions [11]. Structured quizzes and a personal progress dashboard accompany each simulation as supporting features.

The platform is also curriculum-aware, surfacing simulations relevant to what a student is currently studying rather than presenting a generic catalog and leaving them to navigate it alone [5]. This is a small feature in implementation terms and a meaningful one in practice.

The objectives of this work are as follows.

1. To develop a web-based simulation platform supporting step-by-step interactive exploration of algorithms and system-level processes across engineering subjects.

2. To implement a curriculum-aware simulation library that surfaces relevant modules based on a student's current academic context.

3. To integrate a per-module quiz mechanism with persistent scoring contributing to a visible progress record.

4. To develop a personal progress dashboard tracking visited modules, quiz performance, cleared module counts, bookmarks, and recent activity.

5. To ensure the platform is secure, cross-device accessible, and deployable without heavy institutional infrastructure dependencies.

II. LITERATURE REVIEW

Simulation-based learning in engineering education has been studied extensively enough that the broad finding that simulations improve conceptual understanding compared to static instruction is not really in question. What the literature reveals, when read carefully, is that the details matter considerably more than the headline result. The review that follows is organized into three areas directly relevant to VisualizeIT's design: the evidence base for simulation-based learning in engineering contexts, the evolution of web-based platforms and virtual laboratories, and the persistent gap between simulation tools and integrated assessment.

A. Simulation-Based Learning in Engineering Education

The case for simulation-based learning in engineering education is reasonably well established. Priyakanth and Krishna [3] found that simulation-driven teaching improved exploration skills and academic performance particularly in courses lacking physical laboratory access. Rutten et al. [2] synthesized results across more than fifty quasi-experimental studies and reported moderate to large effect sizes for conceptual understanding and engagement. Chernikova et al. [10] extended this line of evidence through a meta-analysis in higher education specifically, finding consistent learning gains across STEM disciplines when simulations were embedded in structured instructional contexts. More recently, Banda and Nzabamina [4] reported a large effect size for motivation and academic achievement in a PhET-based intervention, though this was in a context with instructor-led integration

rather than independent student use, which is a meaningful distinction. The broad finding is not really disputed. What the literature is less clear about is which design decisions actually drive the benefit.

B. Web-Based Platforms and Virtual Laboratories

Web-based simulation platforms have expanded access considerably, removing the hardware and installation dependencies that constrained earlier desktop tools. Samanthula et al. [5] introduced a cloud-based learning system with role-based access and basic learning analytics, demonstrating that scalable deployment and some degree of progress monitoring were achievable together. Jovanovic et al. [11] developed an interactive browser-based simulation environment for compiler concepts, showing that step-by-step algorithm execution delivered through a web interface could meaningfully improve student understanding of abstract topics. Belec et al. [6] and Negahban et al. [13] both document the broader shift from physical experimentation to digital simulation environments, noting improvements in accessibility while acknowledging that the transition has not resolved questions about how students verify their own learning. Tian et al. [7] found that interactive simulation software in engineering courses improved student engagement and conceptual understanding beyond what traditional instruction alone achieved.

C. Assessment Integration and the Persistent Gap

The limitation that appears most consistently across the literature is the separation between simulation and assessment. Blake and Scanlon [8] argue that simulation effectiveness depends heavily on how well the tool is embedded in a learning process, and that tools used in isolation underperform their potential systematically. Kaldaras and Wieman [9] show that simulations alone do not guarantee deep understanding and that structured scaffolding and feedback are necessary for meaningful learning outcomes. Jovanovic et al. [12] found in the context of algorithm and automata courses that structured step-by-step simulation engagement improved student understanding of abstract concepts compared to passive observation alone. Navarro-Parra and Chiappe [14] conducted a systematic review of simulated learning environments and found that assessment integration remained the most consistently underdeveloped feature across platforms. Xu et al. [15] identify closing the loop between student interaction and self-evaluation as a central open problem for future simulation platforms.

Most platforms evaluated in the literature are assessed through externally administered pre and post-tests. The platforms themselves offer students no internal mechanism to gauge understanding. This is the gap VisualizeIT is designed to address directly.

III. METHODOLOGY

VisualizeIT is a curriculum-aligned, web-based simulation platform built for engineering students. This section describes the system structure, module design, and key development decisions.

A. System Overview

VisualizeIT is designed as a full-stack web application following a client-server architecture. The frontend handles simulation rendering, user interaction, quiz presentation, and dashboard visualization. The backend manages authentication, session handling, data persistence, and business logic. A PostgreSQL database layer, managed through Supabase, stores all user-specific data including profiles, quiz results, bookmarked modules, and activity logs.

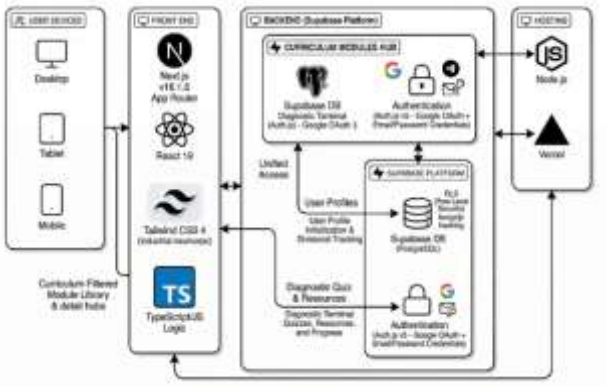

Figure 1 shows the system architecture. The frontend is built with Next.js 16.1.6 (App Router), React 19, Tailwind CSS 4, and TypeScript, delivering a curriculum-filtered simulation library and individual module detail hubs to user devices across desktop, tablet, and mobile. The backend is organized into two primary functional areas within the Supabase platform. The Curriculum Modules Hub handles the simulation catalog, learning resources, and the Diagnostic Terminal quiz mechanism, with authentication managed through Auth.js v5

supporting both Google OAuth and email/password credentials. The Supabase Platform layer manages the PostgreSQL database with Row Level Security (RLS) policies and bcryptjs password hashing, ensuring each user can only access their own records and that no plaintext credentials are stored at any point. The platform is deployed on Vercel, which integrates directly with the Next.js build pipeline and provides edge delivery without requiring separate infrastructure configuration.

A detail worth noting is that simulations execute entirely within the browser rather than being served as rendered output from the backend. This keeps step-by-step interaction fast, avoids server round-trips during execution, and means the backend's role during simulation is limited to logging activity and persisting bookmarks rather than driving the simulation itself.

B. System Modules

The platform is organized into five functional modules, each responsible for a distinct area of the student experience.

The Authentication Module supports two login pathways: Google OAuth via Auth.js v5 for users who prefer single-click access, and email and password credentials with bcryptjs password hashing for users who prefer an independent identity. Supabase Row Level Security policies ensure that each user can only read and modify their own records.

The Onboarding Module intercepts first-time users after authentication and routes them to a profile initialization form that collects branch, academic year, semester, and division. This data drives the curriculum filter in the simulation library and the division-specific tracking in the dashboard.

The Simulation Library reads the authenticated user's semester and branch from their profile and filters the available simulation catalog to surface modules relevant to their current coursework. An override toggle allows students to browse the full catalog if they choose. Each simulation entry is configured through a JSON data file, which means adding new simulations does not require changes to the application logic.

Each simulation has a dedicated **Detail Hub** containing the simulation launcher, a structured list of learning objectives, curated supplemental resources, and a Diagnostic Terminal that presents a quiz. Passing the quiz with 60% or above marks the module as Cleared in the user's progress record.

The Progress Dashboard provides a personal view of academic activity. It displays exploration metrics showing visited versus unvisited modules, a Mastery Index showing quiz accuracy and cleared module counts, a bookmark stack for frequently used simulations, and a log of recent session activity.

C. Development Methodology

Requirements were derived from two sources: the limitations documented in the simulation-based learning literature, and direct familiarity with the curriculum needs of the target student cohort. System design preceded implementation. The module boundaries, data schemas, API routes, and authentication flows were planned before code was written. Frontend and backend development proceeded in parallel across team members, with integration occurring after individual modules reached functional stability. Testing covered authentication flows, quiz submission and scoring, progress record updates, and simulation state management across multiple browser environments.

The platform was built module by module, with each component tested independently before integration. The frontend was developed using Next.js and React, structured around the App Router convention. Tailwind CSS handled styling throughout. TypeScript was used across the entire codebase, which caught a meaningful number of type-related issues during development that would otherwise have appeared at runtime.

Simulations execute entirely within the browser. Each simulation maintains its own state object that updates on every user interaction, whether stepping forward, stepping back, or changing an input. An early attempt at a shared state structure across all simulation types introduced inconsistencies, because what counts as a meaningful step in a sorting algorithm differs enough from a CPU scheduling simulation that a single abstraction could not serve both cleanly. Each simulation was given its own schema, which meant more code but considerably

fewer rendering issues during testing. The simulation catalog is defined in JSON configuration files separate from the application code, so adding or updating modules requires no code changes. The complete application is deployed on Vercel with continuous deployment configured from the main branch.

IV. RESULTS AND DISCUSSION

The system was evaluated through functional testing across all major user flows: registration via Google OAuth and email credentials, profile onboarding, simulation browsing with the curriculum filter active and overridden, step-by-step and auto-execution modes, quiz submission and scoring, Cleared status assignment, and dashboard data accuracy. All flows performed correctly and stably across desktop and mobile environments.

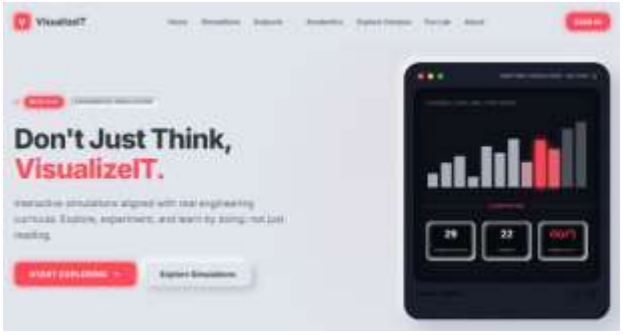

Figure 2: Landing Page of VisualizeIT

Figure 2 shows the platform landing page. The hero section embeds a live Bubble Sort preview mid-execution, with comparisons, swaps, and complexity displayed as live metrics. A prospective user sees the platform in action before logging in, which communicates its interactive nature more directly than descriptive text could. The navigation bar exposes the subject-based structure of the simulation library, reflecting the curriculum-aware filtering that activates once a user's profile is configured.

Figure 3 shows the Sorting Visualizer during a Bubble Sort run. The state panel above the visualization exposes the pass counter, current i and j index values, and cumulative swap count at each step. The pseudocode panel on the right highlights the currently executing line in sync with the visual, connecting algorithm logic to behavior directly. A

Figure 3: Sorting Visualizer for DSA

student observing both simultaneously is better positioned to transfer that understanding to a written or practical examination than one watching the animation alone.

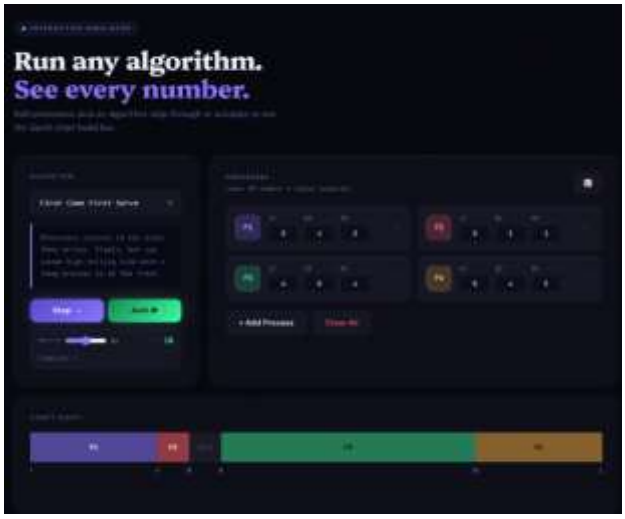

Figure 4: CPU Scheduling Simulator for OS

Figure 4 shows the CPU Scheduling simulator following a completed FCFS run on four processes. The per-process results table displays arrival time, burst time, completion time, turnaround time, and waiting time for each process. The calculations panel shows average waiting time, turnaround time, throughput, and CPU utilization alongside the formulas used to derive them. Modifying a single process's burst time and re-running the simulation updates every derived metric immediately, giving students direct intuition for how scheduling decisions propagate through the system.

The platform was tested across Chrome, Firefox, and Safari on desktop, and on mobile browsers on Android and iOS. No significant rendering issues were observed. Full-cohort simultaneous load was not tested and remains an open item.

The quiz mechanism in its current form is an instrument for measuring conceptual understanding. Multiple-choice questions can confirm surface recall but cannot reliably detect whether a student understands why an algorithm behaves as it does. This is a known limitation of the current design and shapes what can reasonably be claimed from the evaluation results.

The platform was demonstrated to over fifty undergraduate students across multiple classroom sessions. Feedback from open-ended discussion was consistently positive. Students found the step-by-step interaction more useful than static textbook representations, and faculty noted that curriculum-aligned filtering addressed a navigation difficulty they had seen students encounter with general-purpose tools. These observations support the platform achieving its design intent and motivate the formal comparative study proposed in future work.

V. CONCLUSION

VisualizeIT was built around a straightforward observation: certain engineering concepts are genuinely difficult to understand from static slides and blackboard diagrams alone, not because students are inattentive but because the behavior those concepts describe unfolds over time and cannot be captured in a still image. The platform addresses this directly through custom interactive simulations that let students manipulate inputs, step through execution, and observe system behavior for themselves. Curriculum alignment, diagnostic quizzes, resource linking, and progress tracking accompany the simulations as supporting features that make the platform more useful in practice. The system is functionally complete, stable, and accessible across desktop and mobile environments. Whether it produces measurable improvements in examination outcomes is a question a formal comparative study must answer, and the current evaluation does not claim otherwise. What can be said is that the platform makes the kind of interaction that was previously unavailable in a curriculum-aligned, assessment-integrated form accessible in a single deployable web application, and that student and faculty reception was encouraging. The system is ready for broader deployment and formal evaluation.

REFERENCES

[1] R. Lis, "Role of visualization in engineering education," Advances in Science and Technology Research Journal, vol. 8, no. 24, pp. 111-118, 2014. doi: 10.12913/22998624/567.

[2] N. Rutten, W. R. van Joolingen, and J. T. van der Veen, "The learning effects of computer simulations in science education," Computers & Education, vol. 58, no. 1, pp. 136-153, 2012. doi: 10.1016/j.compedu.2011.07.017.

[3] R. Priyakanth and N. M. S. Krishna, "Impact of simulation-based teaching in the development of students' exploration and learning skills," Journal of Engineering Education Transformations, vol. 36, no. 3, pp. 45-52, 2023. doi: 10.16920/jeet/2023/v36i3/23008.

[4] H. J. Banda and J. Nzabamina, "The impact of physics education technology (PhET) interactive simulation-based learning on motivation and academic achievement among Malawian physics students," Journal of Science Education and Technology, vol. 32, no. 4, pp. 542-558, 2023. doi: 10.1007/s10956-023-10045-3.

[5] B. K. Samanthula, M. Mehran, M. Zhu, N. Panorkou, and P. Lal, "Experiences toward an interactive cloud-based learning system for STEM education," in Proc. IEEE Integrated STEM Education Conference (ISEC), 2020, pp. 1-8. doi: 10.1109/ISEC50023.2020.9280738.

[6] J. Belec, A. Salman, and K. Bakr, "Simulation in engineering education: The transition from physical experimentation to digital immersive simulated environments," Computers & Education, vol. 146, p. 103734, 2020. doi: 10.1016/j.compedu.2019.103734.

[7] C. Tian, C. Zhang, S. Liu, J. Zhang, X. Liu, and B. Zhu, "Incorporating scientific applications into engineering education through interactive simulation software," Computer Applications in Engineering Education, 2025. doi: 10.1002/cae.70057.

[8] C. Blake and E. Scanlon, "Reconsidering simulations in science education at a distance: Features of effective use," Journal of Computer Assisted Learning, vol. 23, no. 6, pp. 491-502, 2007. doi: 10.1111/j.1365-2729.2007.00239.x.

[9] L. Kaldaras and C. Wieman, "Employing technology-enhanced feedback and scaffolding to support the development of deep science understanding using computer simulations," International Journal of STEM Education, vol. 11, no. 1, pp. 1-18, 2024. doi: 10.1186/s40594-024-00490-7.

[10] O. Chernikova, N. Heitzmann, M. Stadler, D. Holzberger, T. Seidel, and F. Fischer, "Simulation-based learning in higher education: A meta-analysis," Review of Educational Research, vol. 90, no. 4, pp. 499-541, 2020. doi: 10.3102/0034654320933544.

[11] N. Jovanovic, S. Stamenkovic, D. Miljkovic, Z. Jovanovic, and P. Chakraborty, "ComVIS: Interactive simulation environment for compiler learning," Computer Applications in Engineering Education, vol. 30, no. 1, pp. 1-15, 2022. doi: 10.1002/cae.22456.

[12] N. Jovanovic, S. Stamenkovic, and P. Chakraborty, "Teaching concepts related to finite automata using ComVis," Computer Applications in Engineering Education, vol. 29, no. 1, pp. 1-15, 2020. doi: 10.1002/cae.22353.

[13] A. Negahban, S. Ozden, and O. Ashour, "Simulation in engineering education: The transition from physical experimentation to digital immersive simulated environments," SIMULATION: Transactions of the Society for Modeling and Simulation International, 2024. doi: 10.1177/00375497241229757.

[14] S. L. Navarro-Parra and A. Chiappe, "Simulated learning environments as an interdisciplinary option for vocational training: A systematic review," Simulation & Gaming, vol. 55, no. 2, pp. 1-22, 2024. doi: 10.1177/10468781231221904.

[15] Y. Xu, J. Sun, and J. Peng, "Present and future trends of virtual simulation in education: A bibliometric analysis," 2025. doi: 10.1177/01678329251329554.