PAIDEIA LMS: A MODERN LEARNING MANAGEMENT SYSTEM

Introducing Paideia LMS: A modern, AI-native alternative to Moodle and Canvas designed to lower administration and hosting costs.

KEN YEUNG

FALL 2025 WHITEPAPER

SUMMARY

SUMMARY

SUMMARY

EXECUTIVE SUMMARY

Paideia officially embarked on its development journey on September 29, 2025, and has rapidly progressed to version 0.5, which was first launched on November 3, 2025. In just over a month of development, we have delivered a demonstrable learning management system that represents a paradigm shift in educational technology. Paideia LMS is designed as a modern alternative to legacy platforms like Moodle and Canvas, addressing fundamental challenges of cost, complexity, and technical debt that have plagued the LMS industry.

Built with contemporary technologies including TypeScript, Bun, React, PostgreSQL, and S3-compatible storage, Paideia leverages modern development practices, containerization, and a single-binary deployment model to eliminate complexity and cost barriers. The platform targets medium to large educational organizations with 1,000 to 50,000 users, optimizing for the specific needs and scale of these institutions.

Version 0.5 demonstrates significant progress across multiple domains. Core platform infrastructure is complete, including a fully functional demo instance at demo.paideialms.com with sandbox mode for testing. The dual-mode binary architecture provides both web server and command-line interface capabilities, with comprehensive database migration management. User management supports five distinct roles with password-based authentication and admin impersonation. Course management is fully functional with enrollment, group organization, and hierarchical content structure. Learning activity modules include functional page, quiz, and assignment modules, with discussion module in development. Content management provides basic note-taking, blog functionality, and rich text editing capabilities. The gradebook structure is established, though grading mechanisms are pending implementation.

Paideia operates under clear operational principles: open source and free forever under AGPL-3.0, architectural simplicity with single PostgreSQL database and monolithic design, transparency through public changelogs and quarterly whitepapers, and community-driven development. Design principles emphasize balanced experience design across teaching, learning, and administration; balanced feature design balancing minimalism, battery-included features, and flexibility; and a modern technology stack built on proven, sector-leading technologies.

The platform's value proposition is compelling: 50-80% cost reduction compared to alternatives, deployment in minutes rather than hours or days, minimal infrastructure requirements (2 CPU, 2GB RAM, 40GB disk at approximately 5 USD per month), complete data ownership, and horizontal scalability. Security and compliance considerations are addressed with institutional control over data, FERPA/GDPR readiness, and a commitment to formal security audits as the project matures.

Short-term vision focuses on completing version 1.0 with core activity modules completion, advanced gradebook features, logging and event handling, system administration capabilities, integration and API development, user experience enhancements, and the Drive feature for comprehensive file management. Long-term vision encompasses enterprise integrations, AI and automation capabilities, advanced platform features, and version control and collaboration systems.

> ℹ️ Paideia will remain free software forever, ensuring accessible education technology for all institutions.

WHAT PAIDEIA IS

Paideia LMS represents a paradigm shift in learning management systems, designed as a modern alternative to legacy platforms like Moodle and Canvas. The LMS industry has long been dominated by platforms that are either prohibitively expensive, technically outdated, or operationally complex. Moodle, while open source, requires significant technical expertise and resources to deploy and maintain. Canvas, despite its polished interface, comes with substantial licensing costs that many educational institutions cannot afford.

Paideia stands apart by addressing these fundamental challenges. Built with contemporary technologies and AI-native principles, Paideia aims to significantly reduce both administration and hosting costs while delivering a superior educational experience. Our mission is simple: create a learning management system that is **free forever**, technically superior, and operationally efficient. By leveraging modern development practices, containerization, and a single-binary deployment model, Paideia eliminates the complexity and cost barriers that have plagued traditional LMS platforms.

Unlike legacy systems that require extensive customization and ongoing maintenance, Paideia is designed to be immediately usable while remaining flexible enough to meet diverse institutional needs. We position Paideia not as another incremental improvement, but as a fundamental reimagining of what a learning management system can be when built from the ground up with modern principles, open source values, and educational excellence in mind.

Paideia is not trying to target millions of users. It targets medium to large educational organizations which have more than thousands to tens of thousands of users, typically ranging from approximately 1,000 to 50,000 users. This focused approach allows Paideia to optimize for the specific needs and scale of these institutions, ensuring optimal performance and usability within this range.

Why Paideia: Comparison Overview

The following table provides a comprehensive comparison of Paideia with established LMS platforms, highlighting key differentiators that make Paideia an attractive alternative for educational institutions:

Feature	Paideia	Moodle	Canvas
Licensing and Cost	Free forever (AGPL-3.0)	Free (GPL)	Proprietary - 50,000+ USD/year
Deployment Model	Single binary - minutes	Multi-service - hours/days	Cloud-hosted SaaS
Infrastructure Requirements	2 CPU, 2GB RAM, 40GB disk (approximately 5 USD/month)	Complex multi-service setup	Managed cloud service
Technology Stack	Modern (TypeScript, Bun, React, PostgreSQL)	Legacy (PHP, MySQL)	Modern but proprietary
Architecture	Monolithic, stateless	Monolithic with plugins	Microservices-based
Plugin System	No plugins - batteries-included	Extensive plugin ecosystem	Limited plugin marketplace
Maintenance Complexity	Minimal - single binary updates	High - requires technical expertise	Managed by vendor
Data Ownership	Complete institutional control	Complete institutional control	Vendor-controlled cloud
Deployment Time	Minutes	Hours to days	N/A - SaaS
Update Process	Simple binary replacement	Complex migration procedures	Vendor-managed updates
Support Model	Community-driven	Community or paid support	Vendor support (paid)

Total Cost of Ownership	50-80% lower than alternatives	High (outsourcing + hosting + admin)	High (licensing + per-user fees)
Scalability	Horizontal scaling with load balancer	Vertical scaling required	Vendor-managed scaling
Security and Compliance	Institutional control, FERPA/GDPR ready	Institutional control	Vendor-managed compliance
Customization	API-first, extensible	Plugin-based customization	Limited customization
Target Scale	1,000 - 50,000 users	Unlimited[1]	Unlimited
Development Status	v0.5 - Early stage	Mature (20+ years)	Mature (15+ years)

This comparison demonstrates that Paideia offers a unique combination of cost-effectiveness, operational simplicity, and modern architecture that addresses the fundamental limitations of existing LMS platforms. While Moodle provides open source flexibility, it requires significant technical resources and often necessitates outsourcing. Canvas offers polished interfaces but at substantial licensing costs. Paideia bridges this gap by delivering modern technology with zero licensing costs and minimal operational complexity.

---

[1]Neither Moodle nor Paideia imposes a hard cap on the number of users that can be registered in the system. However, the actual capacity for both platforms is constrained by server infrastructure and resources. In practice, Moodle's scalability characteristics are similar to Paideia, with capacity determined by available hardware, database performance, and system architecture rather than software-imposed limitations.

OPERATIONAL PRINCIPLES

Paideia's operational principles guide how the core team manages the project and provide the vision that directs decision-making. These principles ensure reliability, efficiency, and long-term sustainability of both the platform and the project itself.

Open Source and Free Forever

Open source and free forever is a fundamental operational principle. Paideia is open source and free under the GNU Affero General Public License v3.0 (AGPL-3.0) — forever. This license ensures that Paideia remains freely accessible to educational institutions while maintaining the principles of open source software. What this means is that Paideia can be used, modified, and distributed freely. Schools and educational institutions can host and use Paideia without restrictions. However, if you modify and host this software, you must share your changes with the community. All forks and derivatives must also use the AGPL-3.0 license, ensuring that the open source nature of Paideia is preserved.

Since Paideia strives to be free and open source forever, its development will be completely sustained by community contribution, donation, and sponsorship. This ensures that the platform remains independent and accessible to all educational institutions, regardless of their financial resources.

⚠️ Commercial Hosting Restrictions: While the license permits commercial use, we strongly discourage organizations from hosting this software for schools as a paid service. The spirit of this project is to remain freely accessible to educational institutions.

If you're interested in commercial partnerships or support services that align with our mission, please reach out to discuss collaboration opportunities.

Architectural Simplicity

Since Paideia is designed for medium to large educational institutions rather than massive-scale public SaaS deployments, this fundamentally shapes its operational and design principles. Paideia employs a single central PostgreSQL database rather than distributed database architectures, reducing complexity and operational overhead. The platform uses a monolithic architecture instead of microservices, which simplifies deployment, maintenance, and debugging while reducing the cognitive load on system administrators.

Paideia requires a traditional server-based deployment rather than a serverless architecture. This approach diverges from contemporary web development practices that favor serverless hosting for rapid deployment, but it aligns with the needs of educational institutions that maintain dedicated IT infrastructure and technical staff. The server-based model provides institutions with superior control, operational predictability, and cost efficiency compared to cloud-hosted alternatives.

The server-based deployment model directly addresses the security and data sovereignty requirements of educational institutions. Organizations can deploy Paideia with minimal configuration through a single binary distribution, reducing deployment complexity and the potential for human error. This simplified architecture minimizes the cybersecurity attack surface while enabling institutions to maintain complete data ownership and control. Educational institutions can ensure compliance with privacy regulations such as FERPA, GDPR, and institutional data policies without depending on third-party cloud service providers.

In an era where AI and software capabilities are advancing rapidly, Paideia adopts a focused feature strategy rather than attempting to incorporate every possible LMS capability. The platform will implement essential LMS features to a high standard of quality and reliability. Unlike other LMS platforms that pursue unbounded feature expansion, Paideia has a defined feature scope with a clear completion target. Once core functionality is achieved, development priorities will shift from feature addition to refinement, stability optimization, and performance enhancement. This bounded scope and architectural simplicity enable Paideia to be sustained by a small core development team rather than requiring organizational expansion. This approach ensures long-term project sustainability while maintaining the quality, reliability, and maintainability that educational institutions require.

Transparency and Openness

Transparency and openness are fundamental to how Paideia operates. All changes to Paideia LMS are documented publicly on GitHub and in the documentation. We maintain a comprehensive changelog that tracks every feature addition, bug fix, and improvement made to the platform, ensuring complete transparency about what's changing and when. The changelog is available in multiple places: GitHub Releases include detailed release notes for every release, the documentation site integrates the changelog, and individual changes are tracked in GitHub commit history. This transparency helps users understand what's new, what's been fixed, and what might affect their installations, while also helping the community track the project's progress and plan for updates.

Every three months, we publish a whitepaper on the documentation site that covers progress, vision, and direction. These quarterly whitepapers provide a comprehensive look at where Paideia is, where it's been, and where it's going. Each whitepaper includes a progress recap with a detailed summary of accomplishments in the past quarter, covering new features and capabilities, performance improvements, bug fixes and stability enhancements, community contributions and engagement, and technical milestones and achievements. The whitepapers also outline our long-term vision, showing where we want to take the platform in the next 2-5 years, including major features and capabilities we're planning, architectural improvements and scalability goals, and community growth and engagement objectives. Additionally, each whitepaper presents our short-term vision with immediate plans and near-term objectives, detailing what we're working on in the next quarter, upcoming features and improvements, priority areas for development, and immediate goals and milestones. These whitepapers ensure the community is always informed about Paideia's direction and can plan accordingly, while also providing an opportunity for feedback and discussion about the project's future.

Community-Driven Development

Development is driven by issues and discussions, with decisions made by the core team. We believe in transparent, community-involved development where ideas can be discussed openly before implementation. Anyone can open an issue on GitHub to report bugs, suggest features, or request improvements. Complex topics are discussed in GitHub Discussions, where the community can share ideas and provide feedback. The core team reviews issues and discussions, makes technical decisions, and prioritizes work. Once a decision is made, the work is implemented and documented. We actively encourage community participation through bug reports, feature requests, discussions, and contributions via pull requests for bug fixes and features. While community input is valuable, the core team makes final decisions about technical architecture and design choices, feature prioritization and roadmap, security and stability concerns, and project direction and strategy. This balance ensures that Paideia moves forward with clear direction while remaining responsive to community needs.

All of these operating principles work together to ensure that Paideia LMS development is transparent, community-driven, and sustainable. We believe that open source projects thrive when they're open about their processes, decisions, and direction. Whether you're a user, contributor, or just curious about the project, you can always find out what's happening, what's planned, and how decisions are made. This transparency builds trust and encourages community involvement.

Community Support and Training

Support and training for Paideia operates on a community-driven model, distinguishing it from commercial LMS providers such as Moodle and Instructure, which maintain dedicated support teams. Paideia is a community project founded by Yeung Man Lung Ken, a learning technology specialist and Moodle administrator at Trinity Western University, Canada. As an open source project rather than a commercial enterprise, Paideia depends on its community for support delivery and knowledge dissemination.

As the project matures and the community expands, support responsiveness will improve significantly as common questions become documented and searchable. Support channels are structured through GitHub Issues for bug reports and technical problems, and GitHub Discussions for complex topics and feature requests. The community actively participates in peer-to-peer assistance, establishing a self-sustaining support ecosystem that scales with adoption.

Paideia will provide comprehensive training resources including instructional video content on YouTube and extensive documentation. The documentation is maintained at https://docs.paideialms.com and is continuously updated to reflect platform evolution. Documentation and training materials are designed to be AI-native, enabling users to leverage AI-powered search and assistance tools to find answers, identify similar issues, and access contextual help.

During the early stages of Paideia, the core development team may provide professional support, training resources, and onboarding assistance. However, this model is not sustainable as the community scales, because Paideia maintains a small core team to preserve operational efficiency and focus on platform development rather than support operations. As the community grows, the support model will transition to community-driven assistance, with the core team concentrating on platform development and the community assuming responsibility for user support and training.

DESIGN PRINCIPLES

Paideia LMS is built on core design principles that guide our development and ensure the platform delivers on its promises. These principles explain why we built Paideia, the problems it solves, and why Paideia may be the best choice for your institution.

Balanced Experience Design

Paideia is committed to maintaining a balance between teaching experience, learning experience, and administration experience. We recognize that a successful learning management system must serve all three constituencies effectively, ensuring that educators can teach efficiently, students can learn effectively, and administrators can manage operations seamlessly.
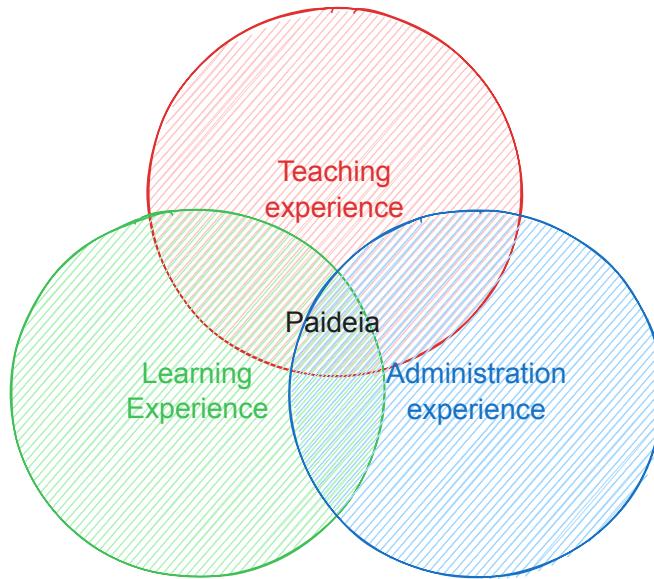
Figure 1 - Paideia's commitment to balancing Teaching Experience, Learning Experience, and Administration Experience

Students prioritize the quality of instructional delivery, emphasizing user experience, system usability, material readability, and interaction with instructors. The learning experience is optimized for clarity, accessibility, and engagement, ensuring that students can focus on learning rather than navigating complex interfaces.

Educators prioritize material creation and student interaction. However, teaching involves multiple roles: instructors who deliver content, teaching assistants who support instruction, course managers who oversee course operations, content managers who curate materials, and instructional designers who design learning experiences. In some institutions, instructional designers collaborate with instructors to create materials. Paideia facilitates cooperation between these roles, enabling seamless collaboration in course development and delivery while maintaining clear interaction channels between instructors and students.

Administrators prioritize ease of self-management, migration from existing systems, upgrade processes, and cybersecurity. These operational requirements are critical for institutional adoption and long-term sustainability.

Paideia addresses all of these requirements and integrates them in a balanced manner, achieving optimal equilibrium where teaching, learning, and administration experiences are optimized simultaneously. This

balanced approach ensures that no single experience is prioritized at the expense of others, creating a harmonious ecosystem where teaching, learning, and administration work together to support educational excellence.

Balanced Feature Design

In designing Paideia's feature set, the platform strives for minimalism and simplicity while simultaneously providing comprehensive, batteries-included functionality out of the box. Additionally, Paideia pursues flexibility, extensibility, and possibilities for institutional adaptation. These three objectives may appear contradictory, but they are harmoniously achieved through careful selection and organization of the feature set.
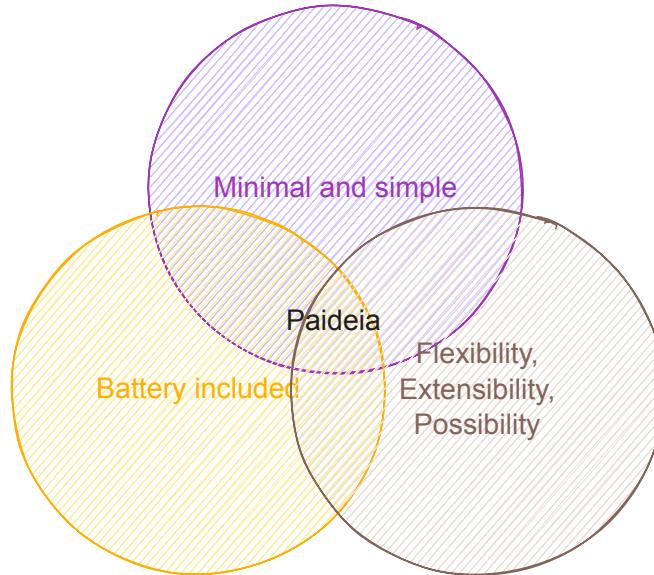

Figure 2 - Paideia's balanced approach to feature design: minimalism, batteries-included features, and flexibility

Applications such as Notion demonstrate that complex applications with extensive functionality can remain accessible when the user interface organizes the feature set exceptionally well. Users never find such systems overly complicated to use because the challenge lies not in having a comprehensive feature set, but in how it is handled. Paideia employs a highly structured and organized approach to the user interface, ensuring that users never find it complicated despite the comprehensive functionality.

This balanced approach means that Paideia includes all essential LMS features out of the box, eliminating the need for plugins or extensive configuration. However, the feature set remains focused and minimal through careful selection, avoiding unnecessary complexity. The structured and organized user interface ensures that even with comprehensive functionality, the system remains intuitive and manageable.

Paideia delivers comprehensive LMS functionality out of the box. Within 30 days of development, we have built a complete Learning Management System with 27 database collections covering all essential LMS features. Paideia includes course management with sections and categories, user management with role-based access control (admin, content-manager, analytics-viewer, instructor, student), activity modules including pages, whiteboards, assignments, quizzes, and discussions, an advanced quiz builder with drag-and-drop interface, a gradebook system with categories and items, enrollment management with groups, media storage with S3 integration, and search functionality. Planned features include LTI (Learning Tools Interoperability) support for seamless integration with external tools, AI-native features built into the platform, and built-in Microsoft integration (Teams, OneDrive, Office 365). Unlike other

platforms that require plugins or extensive configuration, Paideia is feature-complete. Institutions do not need to search for plugins, manage compatibility concerns, or handle complex dependencies.

Paideia avoids the plugin architecture that characterizes Moodle and Canvas. Instead of a plugin system that leads to compatibility issues, management overhead, and security concerns, Paideia includes all essential features built-in. We intentionally limit user customization to reduce support complexity, maintain system stability, ensure consistent user experience, and lower administrative overhead. Traditional LMS platforms with plugin architectures face several challenges: plugin compatibility issues where updates can break plugins, requiring constant maintenance; security vulnerabilities where each plugin represents a potential security risk; management overhead in tracking plugins, updates, and dependencies; performance issues where poorly written plugins can degrade system performance; and vendor lock-in where some plugins are commercial-only, adding costs. Paideia's approach differs fundamentally. We include all essential features built-in, so institutions do not need plugins for core functionality. This approach eliminates compatibility concerns, provides better security through fewer moving parts, simplifies management, ensures consistent performance, and avoids vendor lock-in. For institutions requiring custom functionality, Paideia's modern codebase is designed for extension. Unlike Moodle and Canvas, basic features are all included — no plugins required.

For flexibility, extensibility, and possibilities, Paideia achieves these objectives through multiple approaches. At the integration level, we provide external APIs and webhooks, enabling third-party developers to extend functionality. We follow LTI (Learning Tools Interoperability) standards, allowing seamless integration with external educational tools. We are also committed to building built-in integrations with other systems. Microsoft serves as one example, as we recognize that many educational institutions use Microsoft to handle their authentication, student file storage, and communications. Microsoft is one of the built-in integrations that we are targeting. We are also open to adding other built-in integrations when we identify the necessity. At the code level, we have designed the application to be modular, so dependencies can be switched or added to the project with minimal effort. This modular architecture enables institutions to adapt and extend the system to meet their specific needs when necessary.

In the era of AI, these three values become increasingly important because AI will offload numerous tasks and will become increasingly general and flexible in the types of tasks it can handle. We are prepared to integrate AI into our system, and it will serve as one of the core components to support these values and make integration of these three values possible. AI will enable Paideia to maintain minimalism while providing comprehensive functionality through intelligent automation, and it will enhance flexibility by adapting to diverse institutional needs through AI-powered customization.

Applications such as Notion serve as excellent examples of how comprehensive functionality can coexist with simplicity through careful organization. By achieving this optimal balance, Paideia delivers a platform that is both comprehensive and manageable, both feature-complete and extensible.

TECHNICAL FOUNDATION AND ARCHITECTURE

Paideia was built to solve fundamental problems that plague traditional learning management systems. Educational institutions face mounting costs, complex deployments, security concerns, and platforms that fail to keep pace with modern educational needs. Paideia addresses these challenges through deliberate architectural choices that prioritize reliability, maintainability, and institutional control.
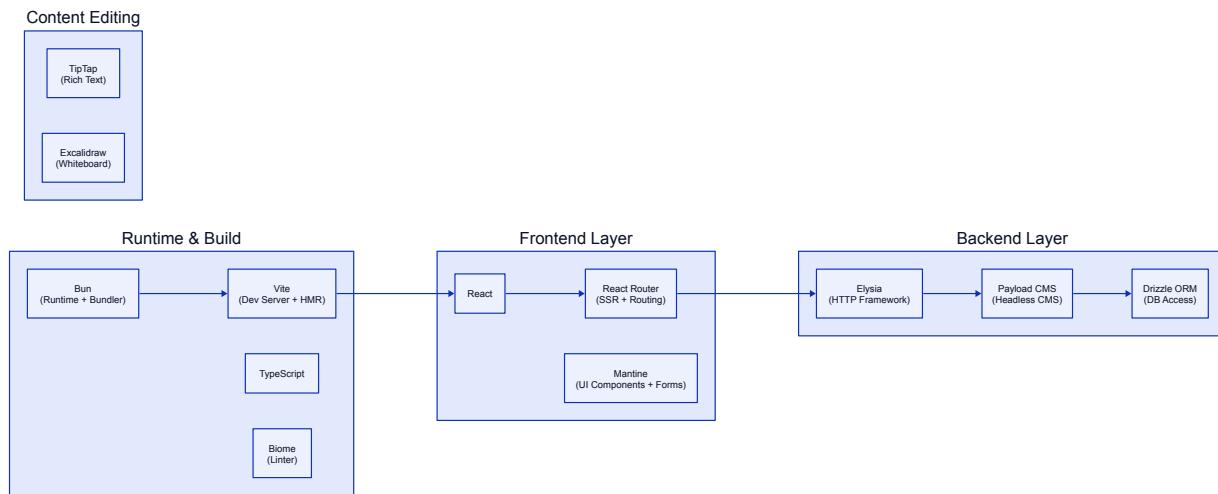

Figure 3 - Paideia's layered architecture: Content Editing, Runtime & Build, Frontend Layer, and Backend Layer

Paideia's architecture is organized into distinct layers, each serving a specific purpose while maintaining clear separation of concerns. This layered architecture ensures that each component can evolve independently while maintaining clear interfaces between layers.

Modern Technology Stack

Paideia is built with cutting-edge technologies and best practices from day one. This means no legacy code, no technical debt, and no need to support outdated patterns. We chose technologies that are performant, maintainable, and have a bright future. By contrast, traditional LMS platforms like Moodle and Canvas were built years ago with older technologies. They've accumulated technical debt over years of development, making them harder to maintain, slower to update, and more complex to deploy. These platforms often struggle with performance issues, security vulnerabilities, and require significant resources to maintain.

Paideia's technology stack is organized into distinct layers. The Runtime & Build layer encompasses TypeScript throughout for comprehensive type safety, Bun as both runtime and bundler providing a simple yet powerful runtime API with exceptional performance and built-in cross-platform compilation capabilities, Vite for development server and hot module replacement, and Biome for code quality. The Backend Layer consists of Elysia as the high-performance HTTP framework, Payload CMS local API to handle the backend, which builds on top of Drizzle ORM, a very powerful TypeScript ORM, to handle most database transactions. The Frontend Layer is built on React as its firm foundation, providing battle-tested reliability and a massive ecosystem, with React Router v7 serving as the lightweight framework that enables composability and flexibility throughout Paideia's architecture through server-side rendering

and routing capabilities, and Mantine delivering comprehensive UI components and form handling. The Content Editing layer provides rich text editing capabilities through TipTap and whiteboard functionality through Excalidraw, enabling educators to create engaging learning materials.

For data storage, Paideia employs PostgreSQL and S3-compatible storage as its foundation, providing a minimal footprint while delivering exceptional scalability. PostgreSQL is one of the most reliable and scalable database systems available, trusted by organizations of all sizes across diverse industries, efficiently handling millions of records. S3-compatible storage provides virtually unlimited storage capacity with industry-standard reliability and durability, scaling to exabyte capacity. These proven solutions are extremely robust and battle-tested, supporting millions of organizations in production operations and scaling exceptionally well. This architectural approach results in a lightweight application layer where Paideia itself has minimal resource requirements, while the underlying infrastructure (PostgreSQL and S3) represents the same production-grade components used in enterprise systems worldwide. Both technologies are production-proven in enterprise environments globally, and the cost-effective consumption model ensures institutions only pay for the resources they consume, with the ability to optimize costs as they scale. Whether serving 100 students or 100,000, Paideia's architecture scales seamlessly with institutional growth. The database and storage layers are designed to handle exponential growth, and Paideia's optimized query patterns ensure consistent performance as data volume increases. All of these technologies are leaders in their respective sectors, performing exceptionally well and establishing industry standards. By leveraging these proven, sector-leading technologies, Paideia can offload significant heavy lifting to them, which keeps the Paideia source code small, effective, and minimizes technical debt. This strategic approach demonstrates that Paideia not only embraces the latest technologies but also follows proven solutions when necessary to build the best possible technology stack.

Production-Grade Architecture

Paideia's architecture is designed with production-grade principles from inception, incorporating enterprise-grade practices throughout the development lifecycle. The platform employs result-based error handling methodologies that ensure predictable and comprehensive error management across all application components. Database operations are transaction-based, guaranteeing data integrity and consistency at every stage of processing. Migration strategies are designed to be non-breaking and backward-compatible, enabling seamless upgrades without service interruption. Rigorous coding standards eliminate type casting in internal functions, maintaining code quality and preventing runtime errors through compile-time type safety.

This architectural foundation ensures that Paideia is reliable, maintainable, and production-ready from initial deployment. Unlike platforms that require extensive hardening phases, Paideia is constructed with enterprise-grade practices from the outset. Data integrity is fundamental to Paideia's design philosophy, and the platform commits to **non-breaking updates forever**, with a **zero data loss guarantee**. All updates can be validated through dry-run upgrade testing prior to deployment, and the platform maintains a strict backward compatibility commitment. This approach enables educational institutions to upgrade with complete confidence, assured that their data and configurations will remain intact throughout the upgrade process.

Operational efficiency is achieved through thoughtful design choices. Each course module exists as a single database record, content is stored in YAML format for human readability and version control, and the system uses PostgreSQL-native version control. These decisions result in minimal resource requirements, making Paideia accessible to institutions with varying levels of technical infrastructure. This technology foundation enables Paideia to deliver a performant, maintainable platform that can evolve with the latest web standards. The focus on modern, sector-leading technologies ensures that institutions receive a platform built for today's needs and tomorrow's opportunities, making Paideia the optimal choice

for institutions seeking a modern, reliable, and maintainable learning management system that prioritizes institutional control and data sovereignty while delivering the performance and features that educators and students require.

Paideia implements a completely stateless architecture where all data is persisted in PostgreSQL and S3-compatible storage, ensuring that no data or state is maintained on the application server itself. All system data—including courses, users, assignments, grades, and media files—is stored in PostgreSQL for structured data and S3-compatible storage for media and file assets.

Browser

HTTP Requests

Bun Process

Frontend Server
Port 3001
reactRouter()

Treaty calls

Cache

API Calls

api
(Treaty Client)

unstorage
(LRU Cache)

Type-safe proxy

getLoadContext()

Backend Server
Port 3000
Elysia API

Direct access

payload
(Payload CMS Instance)

Storage

Drizzle ORM

S3/MinIO
Media Storage

PostgreSQL
Database
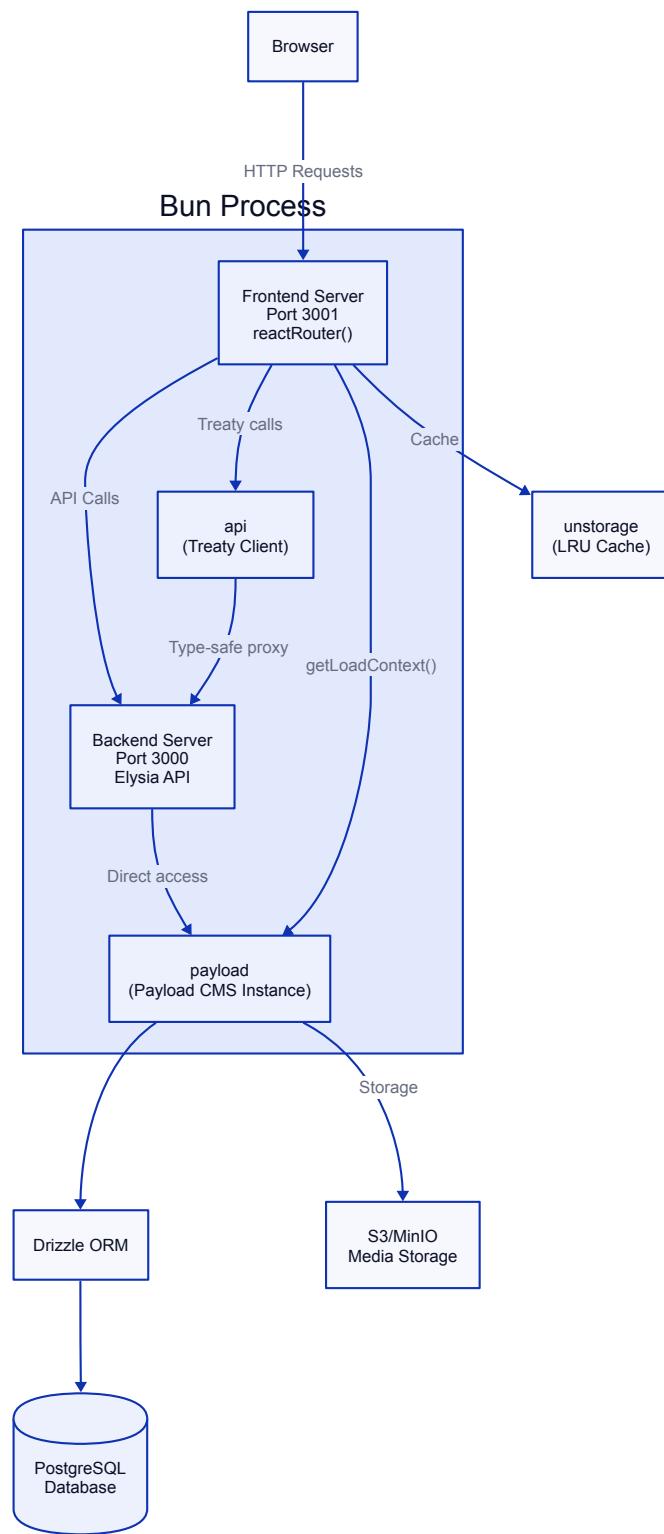
Figure 4 - Paideia's runtime architecture: Browser, Bun Process, Frontend Server, Backend Server, Payload CMS, Drizzle ORM, PostgreSQL, and S3/MinIO

Paideia's runtime architecture demonstrates how the stateless design operates in practice. The browser communicates with the Bun Process, which hosts both the frontend and backend servers as separate processes. The Frontend Server utilizes React Router for server-side rendering and routing capabilities, while the Backend Server provides the Elysia API. The frontend communicates with the backend through

a type-safe client, enabling seamless API integration. The Backend Server directly accesses Payload CMS, which in turn employs Drizzle ORM to interact with PostgreSQL for structured data operations. Payload CMS also manages media storage through S3-compatible storage. The frontend server can access Payload CMS directly for server-side data loading, and caching mechanisms provide enhanced performance.

This dual-server architecture provides several key benefits, with separation of concerns being the primary advantage. By separating frontend and backend concerns, each server maintains distinct responsibilities, creating clear boundaries between presentation and business logic layers. This separation promotes modularity, where each server functions as an independent module that can be developed, tested, and deployed separately. The modular design enhances maintainability by enabling developers to work on frontend and backend components independently without interfering with each other's work.

This stateless architectural approach delivers several significant benefits. It provides exceptional scalability since no data resides on the server, enabling institutions to scale Paideia to millions of users by adding additional application instances, while the database and storage layers handle scaling independently. Migration is straightforward because nothing is stored on the server—institutions simply configure their new Paideia instance to connect to the same PostgreSQL database and S3 storage, eliminating the need for data export/import procedures or complex migration scripts. The stateless design enables high availability through the deployment of multiple Paideia instances behind a load balancer, where if one instance becomes unavailable, other instances continue serving requests seamlessly. Finally, the architecture enables zero downtime deployments where institutions update Paideia by replacing the binary and restarting the service, with no data migration required because the data persists independently in PostgreSQL and S3.
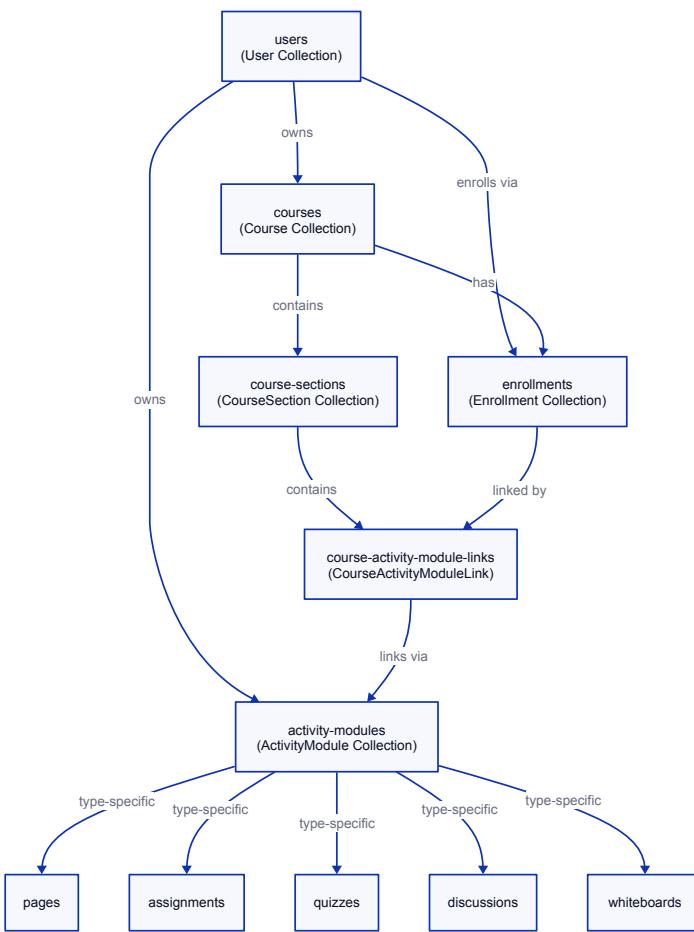
Figure 5 - Paideia's data model: relationships between users, courses, enrollments, sections, and activity modules

Paideia's data model is designed to reflect the natural relationships within educational institutions while maintaining flexibility and extensibility. The core entities include users, courses, enrollments, course sections, activity modules, and their interconnections. Users own courses and enroll in courses through enrollments, creating a many-to-many relationship that supports complex institutional structures. Courses contain course sections, which organize learning content into manageable units. Course sections link to activity modules through course-activity-module-links, enabling flexible content organization.

Activity modules serve as the foundation for various learning activities, with type-specific collections for pages, assignments, quizzes, discussions, and whiteboards. Paideia deliberately chose these five main activity modules based on observations that educational organizations primarily use these activities in their instructional practices. Unlike systems such as Moodle that provide numerous activity modules, most of which experience very low usage and consequently increase maintenance costs and technical debt, Paideia avoids this approach. Instead, each activity module is designed to be feature-complete, enabling it to accommodate diverse institutional needs while maintaining simplicity and avoiding feature bloat. While it is possible that Paideia may add more activity module types in the future, we will maintain the principle of keeping the number of activity modules limited to avoid the feature bloat that plagues other systems. This design allows each activity type to have specialized fields and behaviors while sharing common functionality through the activity-modules collection. The data model supports hierarchical relationships, enabling institutions to structure content according to their pedagogical approaches while maintaining data integrity through referential constraints and transaction-based operations.

Paideia CLI Mode

Paideia LMS is designed as a dual-mode application that functions both as a web server and a command-line interface, all within a single binary. This design eliminates the need for separate administrative tools or complex management interfaces, enabling administrators to perform essential tasks directly from the terminal with simple, intuitive commands.
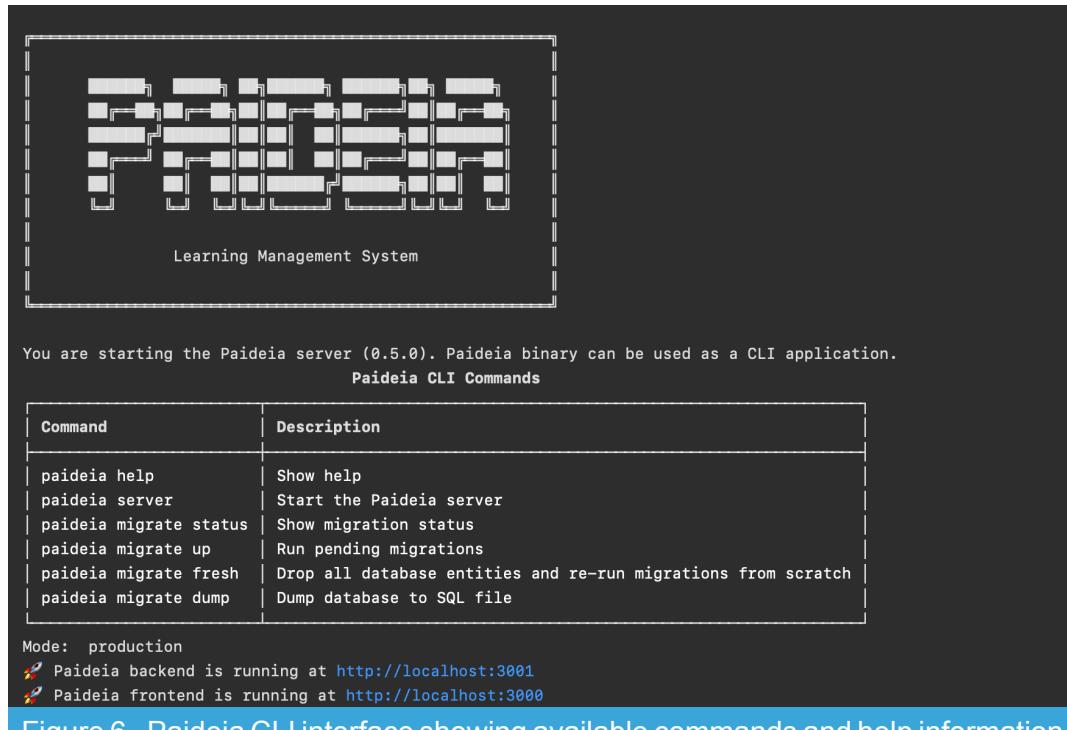

Figure 6 - Paideia CLI interface showing available commands and help information

The command-line interface is complete and provides straightforward administrative capabilities that simplify system management. Administrators can start the Paideia server with a single command, check the status of database migrations, apply database updates, and create backups—all through simple, memorable commands. Additional CLI commands will be added in future versions to expand administrative capabilities. This approach reduces the learning curve for system administrators and eliminates the need for extensive training or specialized knowledge.

Database migration management is complete and available in both the admin panel and CLI mode. Administrators can check migration status to see which database updates have been applied and which are pending, and apply pending migrations through either interface. The CLI also enables database backup creation before making changes. The backup functionality automatically saves database dumps to a dedicated directory, making it easy to maintain regular backups and restore data if needed. These capabilities ensure that database maintenance is straightforward and safe, reducing the risk of data loss during updates or migrations.

The CLI mode works seamlessly with Paideia's single binary deployment model, meaning that all administrative capabilities are available whether running Paideia in development or production environments. This consistency ensures that administrators can use the same commands and procedures across all deployment scenarios, reducing complexity and potential for errors. The dual-mode design demonstrates Paideia's commitment to operational simplicity, enabling institutions to manage their learning management system efficiently without requiring extensive technical expertise or complex tooling.

DEPLOYMENT AND OPERATIONS

Paideia offers multiple deployment options designed to minimize complexity and reduce setup time. The platform can be deployed as a single binary executable, enabling rapid installation across supported platforms. This deployment model, inspired by pioneering open source projects such as PocketBase, eliminates the need for complex service orchestration, dependency management, and multi-component configuration. Administrators can simply download the platform binary for their target architecture (macOS ARM64, Linux ARM64, or Linux x64, windows x64), configure executable permissions, and launch the application.

For institutions seeking an even more streamlined deployment experience, Paideia maintains an official Docker configuration that enables complete system provisioning through a single docker compose file. This containerized deployment solution allows administrators to instantiate the entire Paideia infrastructure—including the PostgreSQL database, MinIO S3-compatible object storage, and the Paideia application—in under one minute. The Docker-based approach abstracts away infrastructure complexity, providing a turnkey solution that requires minimal technical expertise while ensuring consistent, reproducible deployments across different environments.

These deployment options stand in stark contrast to traditional learning management systems that necessitate complex multi-service architectures, extensive configuration management, dependency resolution, version compatibility coordination, and ongoing operational maintenance. Where legacy platforms require hours or days of deployment effort, Paideia enables production-ready installations in minutes. The single binary approach ensures consistent behavior across deployment environments and simplifies updates through straightforward binary replacement, while the Docker configuration provides an additional layer of convenience for containerized infrastructure.

System Requirements

Paideia's minimal system requirements are exceptionally modest, making it accessible to institutions with varying levels of technical infrastructure. The minimum requirements include 2 CPU cores, 2GB RAM, and 40GB disk space, which can be provisioned for approximately five US dollars per month on most cloud hosting providers. This affordability demonstrates Paideia's efficient resource utilization and makes it accessible to educational institutions with limited budgets.

The demo instance at https://demo.paideialms.com is hosted on a Hetzner CPX21 instance with 3 vCPU, 4GB RAM, and 80GB local disk, with daily backups, costing less than 10 US dollars per month. This production deployment demonstrates Paideia's efficient resource usage and operational cost-effectiveness, serving as a practical example of the platform's minimal infrastructure requirements.

The demo instance operates in sandbox mode, which automatically cleans up all data every midnight. This sandbox environment is available for everyone free of charge, enabling students, instructors, and schools to test Paideia's capabilities without any commitment or cost. We believe that this free instance will promote the project and collect valuable feedback from different perspectives, helping to shape Paideia's development based on real-world usage and diverse user needs.[2]

---

[2]Sandbox mode is built into the system and available to everyone. If institutions want to use sandbox mode, they can try it internally within their school.

Deployment Process

The deployment process for Paideia is comprehensively documented on our documentation website at https://docs.paideialms.com. The documentation provides step-by-step setup instructions for both single binary and Docker-based deployments, enabling administrators to deploy Paideia with minimal technical expertise. The deployment process is designed to be straightforward and can be completed in minutes rather than hours or days.

Paideia fosters a supportive community where institutions can help each other with deployment and setup challenges. Through GitHub discussions, documentation, and community engagement, schools can share knowledge, troubleshoot issues, and learn from each other's experiences. This community-driven approach ensures that institutions do not find it challenging to set up servers by themselves, as they have access to a network of peers and resources who can provide guidance and support.

During the early stage of the project, Paideia will also provide direct assistance to schools that are interested in setting up and trying the system. This hands-on support helps institutions evaluate Paideia's capabilities and ensures that early adopters can successfully deploy and test the platform. As the community grows and matures, this direct support will transition to community-driven assistance, with the Paideia team focusing on platform development while the community provides peer support and knowledge sharing.

Migration from Other Systems

Migration from legacy learning management systems such as Moodle and Canvas presents significant challenges due to fundamental architectural differences. These systems use plugin architectures where different plugin configurations create unique data formats, making migration between instances of the same system difficult or impossible. For example, two Moodle instances with different plugin configurations cannot guarantee data integrity when migrating between each other, even within the same system.

Paideia does not currently provide migration tools from Moodle or Canvas. While migration from these systems is challenging due to the fundamental differences in data formats and architectural approaches, Paideia may provide migration tools in the future once other core features have been completed. The conversion from these systems is complex because their data formats differ significantly, and the plugin architecture adds additional complexity and variability to data structures. However, as Paideia matures and core functionality stabilizes, migration capabilities may become a development priority to help institutions transition from legacy systems.

Paideia's architecture provides a significant advantage for data portability. Since Paideia does not use a plugin architecture, every Paideia instance maintains a consistent data format. This guarantees that data can be migrated, imported, exported, and upgraded between any Paideia instances with complete integrity and security. Institutions can confidently move data between Paideia deployments, upgrade systems, or consolidate installations without data loss or compatibility concerns.

VALUE PROPOSITION AND COST ANALYSIS

Paideia delivers significant cost savings compared to traditional learning management systems through its open source model, efficient resource utilization, and simplified deployment architecture. Unlike proprietary platforms such as Canvas that require substantial licensing fees, Paideia eliminates software licensing costs entirely. The minimal infrastructure requirements—2 CPU cores, 2GB RAM, and 40GB disk space—can be provisioned for approximately five US dollars per month, demonstrating exceptional cost efficiency.

Traditional LMS platforms often require extensive infrastructure investments, complex multi-service architectures, and ongoing maintenance costs that can total tens of thousands of dollars annually. Canvas requires substantial licensing fees that can exceed 50,000 US dollars per year for medium-sized institutions, in addition to infrastructure and support costs.

Moodle, while open source, presents a different set of cost challenges. Due to its management complexity —including a PHP codebase with scattered files and a plugin architecture that creates maintenance overhead—many educational organizations choose to outsource Moodle deployment and management to third-party organizations. These outsourcing providers typically charge institutions by active user, and costs can grow exponentially as student populations increase. Additionally, hosting platforms that provide Moodle hosting typically offer only infrastructure services rather than technical support, meaning institutions must still hire dedicated Moodle administrators within their organizations to manage the system, configure plugins, handle updates, and provide technical support. These combined costs—outsourcing fees that scale with user count, hosting costs, and internal administrator salaries—can result in total costs that rival or exceed proprietary solutions.

Paideia's total cost of ownership is dramatically lower than legacy systems. With no licensing fees, minimal infrastructure requirements, and simplified deployment and maintenance, **institutions can confidently reduce costs by 50 to 80 percent compared to traditional LMS platforms**. These savings come from both reduced management costs—eliminating the need for extensive outsourcing or specialized administrators—and reduced hosting costs due to Paideia's efficient resource utilization. The single binary deployment model eliminates the need for complex service orchestration and reduces ongoing operational overhead, further reducing total cost of ownership while providing a modern, technically superior platform.

SECURITY AND COMPLIANCE

Paideia is designed with security and compliance as foundational principles, recognizing that educational institutions handle sensitive student data and must comply with various regulatory requirements. While Paideia is in its early stages and has not yet undergone formal security certifications or audits, the platform is architected to support compliance with major educational data protection regulations.

Data Protection and Privacy

Paideia's architecture supports compliance with key educational data protection regulations including FERPA (Family Educational Rights and Privacy Act) in the United States, GDPR (General Data Protection Regulation) in the European Union, and COPPA (Children's Online Privacy Protection Act) for institutions serving students under 13. The platform's server-based deployment model allows institutions to maintain complete control over their data, ensuring that sensitive student information never leaves institutional infrastructure unless explicitly configured to do so.

The stateless architecture and consistent data format across all Paideia instances enable institutions to implement comprehensive data protection strategies. All data is stored in PostgreSQL for structured data and S3-compatible storage for media files, allowing institutions to leverage their existing database and storage security measures, backup strategies, and disaster recovery procedures.

Backup and Disaster Recovery

Paideia's architecture facilitates straightforward backup and disaster recovery procedures. Since all data is stored in PostgreSQL and S3-compatible storage, institutions can leverage standard database backup tools and S3 backup capabilities. The stateless design means that application-level backups are unnecessary—all critical data resides in the database and storage layers, which can be backed up independently using industry-standard tools and procedures.

Institutions can implement automated backup strategies for PostgreSQL databases and S3 storage, ensuring that data can be recovered in the event of hardware failure, data corruption, or other disasters. The consistent data format across all Paideia instances guarantees that backups can be restored to any Paideia deployment, providing flexibility in disaster recovery scenarios.

Security Architecture

Paideia's security architecture is built on modern, secure technologies and practices. The platform uses TypeScript throughout for type safety, reducing the risk of common programming errors that can lead to security vulnerabilities. The single binary deployment model reduces the attack surface compared to multi-service architectures, while the stateless design eliminates server-side data storage that could be compromised.

While Paideia has not yet undergone formal security audits or certifications, the platform is designed to support institutions' security requirements. As an early-stage project currently at version 0.5, developed initially by Yeung Man Lung Ken, formal security audits and certifications are not yet available. However, as the project matures and the community grows, formal security audits and certifications will definitely become available. Institutions should conduct their own security assessments based on their specific requirements and regulatory obligations.

FEATURE COMPLETENESS AND MATURITY

Paideia v0.5 represents a demonstrable learning management system with core functionality that enables institutions to evaluate the platform's capabilities. The current release includes essential features for user management, course management, enrollment, learning activities, and administration, providing a solid foundation for educational operations.

Production-Ready Features

The v0.5 release includes production-ready features across core LMS functionality. User management is essentially complete, with users able to register, log in, and log out. The system includes five distinct system roles (admin, content-manager, analytics-viewer, instructor, student), providing comprehensive role-based access control. A guest role may be added in the future, but it is not currently on the roadmap. Email verification is not yet implemented, but it will be added in the upcoming version 1.0. In the future, when Microsoft integration is implemented, MFA will be supported.

Activity management features are mostly functional, enabling users to create pages, whiteboards, quizzes, assignments, except discussions—the current discussion preview is a mockup. In the upcoming version 1.0, all five main activity modules will definitely be complete and fully functional.

Course management features are complete, including course creation, organization, and section management. Category management is fully implemented, allowing institutions to organize courses into logical groupings. Enrollment management is complete, supporting course enrollment and participant management. The organization features for course sections and course modules are also complete, providing comprehensive course structure management.

Administrative features include comprehensive layout systems for users, modules, courses, and course content, as well as admin layout capabilities. The platform includes note management, blog functionality, heatmap visualization, and dashboard mockups. Media storage is integrated with S3-compatible storage, and the platform includes LTI support for integration with external tools.

Throughout the development up to version 0.5, Paideia's core design principles have been consistently upheld. The platform maintains its single binary deployment model, preserves the no-plugin architecture, and includes all essential features as battery-included functionality. These principles will continue to guide development as additional features are added leading up to version 1.0.

Known Limitations and Future Development

As a v0.5 release, Paideia has known limitations and areas for future development. The platform is demonstratable rather than fully production-ready for all use cases. The discussion feature, while previewed in version 0.5, is currently a mockup and will be fully completed in version 1.0. Until version 1.0, additional related features will be added to enhance the platform's capabilities and complete the core functionality.

Feature comparison with legacy systems such as Moodle and Canvas reveals that Paideia currently focuses on core LMS functionality rather than the extensive plugin ecosystems available in those platforms. However, Paideia's built-in feature set is designed to cover essential educational needs without requiring plugins for core functionality. As the platform matures toward version 1.0 and beyond, additional features and capabilities will be added based on community priorities and institutional requirements, while

maintaining the core design principles of single binary deployment, no-plugin architecture, and battery-included functionality.

For detailed information about planned features, development priorities, and the project roadmap, institutions can visit the documentation site at https://docs.paideialms.com, where the full roadmap is available for review.

TECHNICAL CAPABILITIES

Performance and Scalability

Paideia's architecture is designed to handle the target scale of medium to large educational organizations with 1,000 to 50,000 users. The platform's stateless design and efficient database queries ensure consistent performance as user bases grow. The use of PostgreSQL, a battle-tested database that handles millions of records in production environments, provides a solid foundation for scalability.

The dual-server architecture allows for horizontal scaling by adding additional application instances behind a load balancer. Since no data is stored on the application servers, scaling is straightforward—simply add more Paideia instances and point them to the same PostgreSQL database and S3 storage. The database and storage layers handle scale independently, allowing institutions to optimize each layer according to their specific needs.
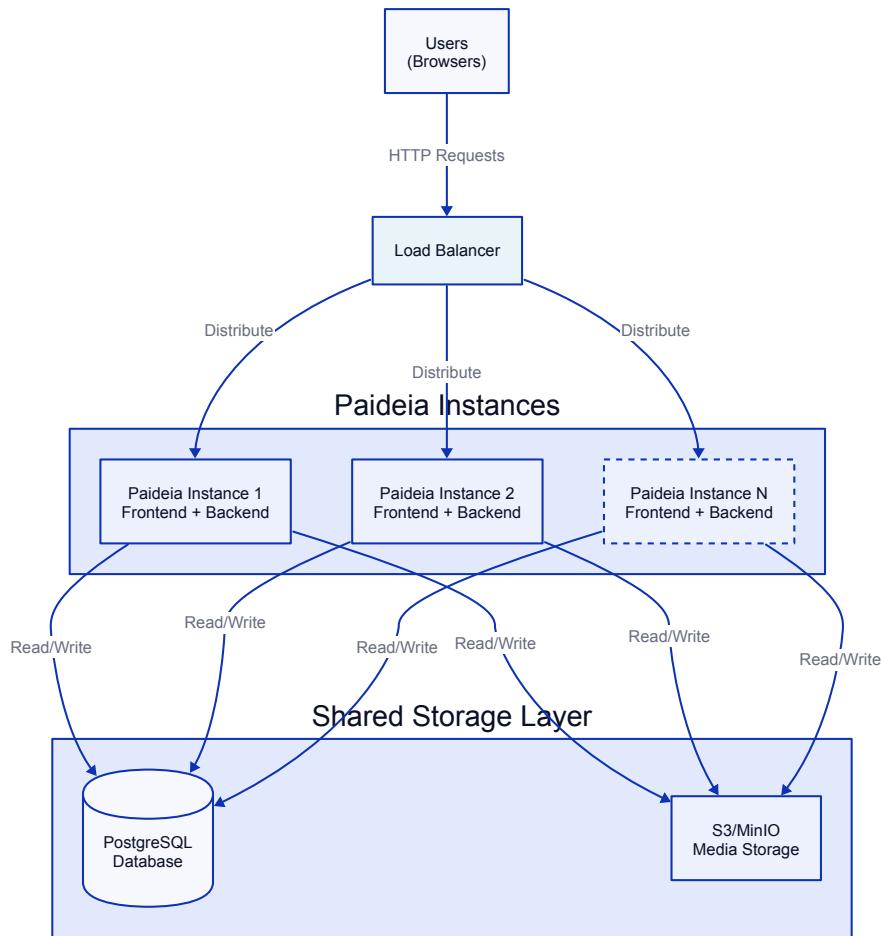

Figure 7 - Paideia's horizontal scaling architecture: Load balancer distributes requests across multiple Paideia instances, all connected to shared PostgreSQL database and S3 storage

Response times are optimized through efficient database queries, caching mechanisms, and the use of modern web technologies. The React Router v7 server-side rendering capabilities provide fast initial page

loads, while the Elysia framework delivers high-performance API responses. As the platform matures, performance benchmarks and optimization metrics will be documented and published.

Integration Capabilities

Paideia will supports integration with external systems through multiple mechanisms. The platform includes built-in LTI (Learning Tools Interoperability) support, enabling seamless integration with external educational tools and content providers. The API-first architecture provides programmatic access to all platform functionality, enabling custom integrations and automation.

Built-in Microsoft integration is planned, recognizing that many educational institutions use Microsoft services for authentication, file storage, and communication. This integration will support Microsoft Teams, OneDrive, and Office 365, providing seamless connectivity with existing institutional infrastructure. Additional built-in integrations may be added based on community needs and institutional demand.

The platform's modern codebase and modular architecture make it straightforward to extend with custom integrations. Webhooks and external API development are supported, allowing institutions to integrate Paideia with student information systems (SIS), single sign-on (SSO) providers, and other institutional systems.

Mobile Access and Responsive Design

Paideia prioritizes responsive design for the web version to ensure mobile accessibility. In version 0.5, development efforts have focused on core functionality rather than dedicated responsive design optimization, given the early stage of the project and limited user base. Nevertheless, the platform benefits from foundational responsive design capabilities through Mantine UI, the component library integrated into Paideia. Mantine components incorporate built-in responsive design features that establish a baseline for mobile accessibility. As the user base expands and core functionality stabilizes, Paideia will prioritize comprehensive mobile responsive design to deliver seamless experiences across all device types.

The platform's React-based architecture establishes a solid foundation for responsive design implementation. As development progresses, the user interface will be systematically optimized to adapt to various screen sizes, ensuring optimal experiences on mobile devices, tablets, and desktop computers. The responsive web interface will deliver full functionality through mobile browsers, enabling students, instructors, and administrators to access Paideia from any device.

Native mobile applications are not currently included in the development roadmap. While mobile application development is technically feasible—particularly given the system's comprehensive API access, which would enable a mobile app to function as a client interface with relatively straightforward technical implementation—it represents a substantial development effort requiring significant resources. The development team has not yet committed to including native mobile applications in the roadmap. However, given that established learning management systems such as Moodle and Canvas provide mobile applications, it is reasonable to expect that a successful Paideia project would likely include mobile application development. Such development is simply not prioritized at this early stage of the project's lifecycle.

Accessibility and WCAG Compliance

Paideia is committed to accessibility and is designed to support WCAG (Web Content Accessibility Guidelines) compliance. The platform uses Mantine components, which include built-in accessibility features and ARIA (Accessible Rich Internet Applications) support. The React-based architecture enables semantic HTML and proper accessibility attributes throughout the user interface.

While Paideia v0.5 has not yet undergone formal WCAG compliance audits, the platform is architected with accessibility in mind. As the project matures, formal accessibility audits and WCAG compliance certifications will be pursued. Institutions should conduct their own accessibility assessments based on their specific requirements and student populations.

REAL-WORLD VALIDATION

Paideia is in its early stages, having begun development on September 29, 2025, and first launched with v0.5 on November 3, 2025. As a new project, the platform does not yet have extensive case studies, user testimonials, or performance benchmarks from production deployments. However, the demo instance at demo.paideialms.com provides a practical demonstration of the platform's capabilities and resource efficiency.

Demo Deployment

The demo instance at demo.paideialms.com is hosted on a Hetzner CPX21 instance with 3 vCPU, 4GB RAM, and 80GB local disk, with daily backups, costing less than 10 US dollars per month. This deployment demonstrates Paideia's efficient resource usage and operational cost-effectiveness, serving as a practical example of the platform's minimal infrastructure requirements.

The demo instance operates in sandbox mode, which automatically cleans up all data every midnight. This sandbox environment is available for everyone free of charge, enabling students, instructors, and schools to test Paideia's capabilities without any commitment or cost. We believe that this free instance will promote the project and collect valuable feedback from different perspectives, helping to shape Paideia's development based on real-world usage and diverse user needs. The demo instance is available for institutions to evaluate Paideia's capabilities and assess its suitability for their needs.

Future Validation

As Paideia matures and gains adoption, case studies from pilot institutions, user testimonials, and performance benchmarks will be documented and published. The community-driven development model ensures that real-world usage will inform platform improvements and feature development. Institutions interested in piloting Paideia are encouraged to engage with the project through GitHub issues and discussions, contributing to the platform's evolution while gaining early access to a modern, cost-effective learning management system.

PROGRESS RECAP

Since development began on September 29, 2025, Paideia has achieved significant milestones leading to the first launch of v0.5 on November 3, 2025. The following progress summary reflects completed features across major functional categories:

⚠ Progress indicators are estimations only and should not be considered accurate measurements. Project requirements are continuously evolving, and additional requirements will be added as development progresses. These progress bars are provided solely to help readers understand the current state of development more easily and should not be interpreted as definitive completion percentages.

Core Platform Infrastructure

Progress: 🟢🟢🟢🟢🟢🟢🟢🟢🟢🟢 (100%)

Target: A production-ready platform infrastructure that provides a demo instance for evaluation, sandbox mode for testing, server monitoring capabilities, and email functionality for system notifications and user communication.

Current State: The demo online version is fully functional and available at demo.paideialms.com, providing institutions with immediate access to evaluate Paideia's capabilities. Sandbox mode is implemented with automatic data cleanup every midnight, enabling users to test the system without concerns about data persistence. Server information and monitoring functionality is complete, allowing administrators to view system status and resource usage. Test email functionality is implemented, enabling administrators to verify email configuration and send test messages.

Remaining Work: Core platform infrastructure is complete and production-ready. All target features have been implemented and are functional.

CLI Mode and Database Migration

Progress: 🟢🟢🟢🟢🟢🟢🟢🟢🟢⬛ (90%)

Target: Paideia operates as a dual-mode binary that functions both as a web server and a command-line interface, enabling administrators to manage the system through both web-based admin panels and terminal commands. The system provides comprehensive database migration management capabilities through both interfaces, allowing administrators to check migration status, apply pending migrations, and perform database backups.

Current State: CLI mode is fully functional and complete. Administrators can start the Paideia server, access help information, check version, and manage database migrations through terminal commands. The CLI provides the following commands: `paideia server` (or simply `paideia`) to start the server, `paideia help` to display available commands, `paideia version` to show version information, `paideia migrate status` to check migration status, `paideia migrate up` to apply pending migrations, `paideia migrate fresh` to reset the database, and `paideia migrate dump` to create database backups.

Database migration management is complete in both the admin panel and CLI mode. In the admin panel, administrators have access to migration management functionality that allows them to view migration status and apply pending migrations through the web interface. In CLI mode, administrators can perform

the same operations through command-line commands, providing flexibility for automated scripts and server management workflows.

Remaining Work: Additional CLI commands will be added in future versions to expand administrative capabilities, but the core CLI functionality and database migration management are complete and production-ready.

User Management System

Progress: ●●●●●●●▇▇▇ (70%)

Target: A comprehensive user management system that supports multiple authentication methods (password and email-based), role-based access control with five distinct roles, user administration capabilities, and multi-factor authentication for enhanced security. The system should enable administrators to manage users, support staff to impersonate users for troubleshooting, and provide secure authentication mechanisms.

Current State: User registration and authentication are functional using password-based authentication only. Users can register with a username and password, log in, and log out. The system includes five distinct system roles: admin, content-manager, analytics-viewer, instructor, and student, each with appropriate permissions. Admin impersonation is fully functional, allowing administrators to view the system from a user's perspective for support purposes. User management and administration capabilities are complete, enabling administrators to create, update, and manage user accounts.

Remaining Work: Email-based registration and login are not yet implemented. Users currently can only register and authenticate using passwords. Multi-factor authentication (MFA) is not yet available but will be implemented when Microsoft integration is completed, as MFA will be supported through Microsoft authentication services.

Course Management
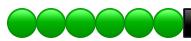
Progress: ●●●●●●●●●● (100%)

Target: A comprehensive course management system that enables instructors to create, organize, and manage courses with sections and categories, enroll students, organize students into groups, and structure course content hierarchically.

Current State: Course creation and organization are fully functional, allowing instructors to create courses, organize them into sections, and structure content hierarchically. Enrollment management is complete, enabling administrators and instructors to enroll students in courses and manage enrollments. Group management and organization are functional, allowing instructors to create groups within courses and assign students to groups. Course categories and structure are implemented, enabling institutions to organize courses into logical groupings.

Remaining Work: Course management is complete and production-ready. All target features have been implemented and are functional.

Learning Activity Modules

Progress: ●●●●●●▇▇▇▇ (60%)

Target: Five fully functional core activity modules (page, whiteboard, quiz, assignment, discussion) that enable instructors to create engaging learning content and students to interact with course materials.

Each module should support complete functionality including content creation, configuration, student submissions, and instructor grading capabilities.

Current State: The page module is fully functional for content delivery, allowing instructors to create rich text content with embedded media. The quiz module is functional with configuration options, enabling instructors to create quizzes with various question types. The assignment module is functional with configuration options, allowing instructors to create assignments with submission requirements. The discussion module exists only as a preview/mockup and is not yet functional—it cannot be used for actual discussions. Course activity management is complete, enabling instructors to organize and manage activities within courses.

Remaining Work: The discussion module needs to be fully implemented with actual functionality. Quiz submission functionality is not yet implemented—students cannot currently submit quiz responses. Discussion submission functionality is not yet implemented—students cannot currently post or reply to discussions. Whiteboard collaboration features need to be implemented. All five activity modules need to support complete submission and grading workflows.

Content Management

Progress: ●●●●●●●●■■ (80%)

Target: A comprehensive content management system that enables users to create, organize, and manage notes, blog posts, and modules with rich text editing capabilities, visualization tools, and social features such as mentioning other users, courses, and activity modules.

Current State: The note management system is functional with basic features complete, allowing users to create, edit, and manage notes. Blog functionality is fully implemented, enabling users to create and publish blog posts. Heatmap visualization is available, providing visual representations of data. Module management is complete, allowing users to create and organize modules. Mermaid diagram support is integrated, enabling users to create diagrams and flowcharts within content.

Remaining Work: The mentioning feature in notes is not yet implemented. Users cannot currently mention other users, courses, or activity modules within notes. This feature would enable social interactions and cross-referencing within the content management system, allowing users to link to other users, courses, or activities directly from their notes.

Assessment and Grading System

Progress: ●●●●■■■■■■ (40%)

Target: A comprehensive grading system that enables instructors to grade assignments, quizzes, and discussions, with flexible grading schemes (number-based and scale-based), feedback capabilities, and comprehensive gradebook reporting. The system should support assignment submission management, real-time grade calculations, and export capabilities for institutional reporting.

Current State: The gradebook system structure is complete with categories and items, enabling instructors to organize grades hierarchically. The grading structure is defined for both number-based grading (numeric scores) and scale-based grading (letter grades or custom scales). The feedback system structure is in place, allowing feedback to be associated with each grade. Gradebook explanation and reporting functionality exists, providing visibility into grading structures. Assignment submission management structure exists, allowing students to submit assignments.

Remaining Work: The grading mechanism is not yet implemented—instructors cannot currently grade assignments, quizzes, or discussions. While the structure exists, the actual grading workflow where

instructors can assign grades, provide feedback, and update student submissions is not functional. Advanced gradebook features such as category totals, column collapsing, rubric-based grading, and course completion tracking are not yet implemented.

User Interface and Layouts

Progress: ●●●●●●●●●● (100%)

Target: A comprehensive user interface system with distinct layouts for different user roles and contexts, providing intuitive navigation and consistent user experience across all platform areas.

Current State: User layout and user module layout are fully implemented, providing students and instructors with dedicated interfaces for viewing and managing their personal content and modules. Course layout is complete with three distinct views: course content layout for viewing course materials, course participants layout for managing enrollments and groups, and course grade layout for viewing grades. Admin layout and dashboard are functional, providing administrators with comprehensive system management capabilities. Comprehensive UI improvements have been implemented, ensuring a polished and consistent user experience throughout the platform.

Remaining Work: User interface and layouts are complete and production-ready. All target features have been implemented and are functional.

Access Control and Security

Progress: ●●●●●●●●■■ (80%)

Target: A comprehensive access control system that enables fine-grained permissions based on user roles, supports administrative functions such as impersonation, and provides secure access management throughout the platform.

Current State: The access control system is fully functional, enabling administrators to manage permissions and access rights throughout the platform. Role-based permissions are implemented for five distinct roles: admin, content-manager, analytics-viewer, instructor, and student, each with appropriate access levels. Admin impersonation is functional, allowing administrators to view the system from a user's perspective for troubleshooting and support purposes.

Remaining Work: Advanced security features such as comprehensive security audits, formal security certifications, and enhanced compliance documentation are not yet available. These will be implemented as the project matures and the community grows.

Overall Platform Progress

Progress: ●●●●●●●■■■ (70%)

Version 0.5 represents a solid foundation for a learning management system, with core infrastructure and content management features complete. While the platform demonstrates significant progress in foundational areas, key functionality such as grading mechanisms, discussion module completion, and public API development remain priorities for version 1.0. The platform is ready for evaluation and testing, establishing a base for continued development toward production readiness.

SHORT-TERM VISION (2025-2026)

Version 1.0 Development Priorities

The immediate focus for Paideia version 1.0 includes completing core functionality, enhancing existing features, and establishing production-ready capabilities:

CORE ACTIVITY MODULES COMPLETION

Progress: ●●●●●●▬▬▬▬ (60%)

Target: Five fully functional core activity modules (page, whiteboard, quiz, assignment, discussion) with complete functionality including content creation, configuration, student submissions, and instructor grading capabilities. All modules should support version control and branching for collaborative development.

Current State: The page module is fully functional for content delivery. The quiz module is functional with configuration options, but quiz submission functionality is not yet implemented—students cannot submit quiz responses. The assignment module is functional with configuration options, and assignment submission structure exists. The discussion module exists only as a preview/mockup and is not functional—students cannot post or reply to discussions. Whiteboard collaboration features are not yet implemented. Activity module version control and branching are not yet implemented.

Remaining Work: The discussion module needs to be fully implemented with actual functionality. Quiz submission functionality must be implemented to allow students to submit quiz responses. Discussion submission functionality must be implemented to allow students to post and reply to discussions. Whiteboard collaboration features need to be implemented. Activity module version control and branching need to be implemented to support collaborative development workflows.

LOGGING AND EVENT MONITORING

Progress: ●●▬▬▬▬▬▬▬▬ (20%)

Target: A comprehensive logging and event monitoring system that stores main event logs in PostgreSQL natively, provides structured querying capabilities for logs and events, and integrates with external systems for further analytics. The system should enable administrators to query and analyze system events, user activities, and platform usage patterns.

Current State: Paideia currently does not have structured logging or event monitoring. All logs are dumped into a single log file without any structured organization. There is no structured way to query logs and events—administrators cannot search, filter, or analyze logs effectively. Main event logs are not stored in PostgreSQL, and there is no integration with external analytics systems.

Remaining Work: A structured logging and event monitoring system must be implemented. Main event logs should be stored in PostgreSQL natively to enable structured querying and analysis. A structured way to query logs and events must be implemented to enable administrators to search, filter, and analyze system events. Integration with external analytics systems must be implemented for further analytics beyond what PostgreSQL can provide. The system should enable administrators to track user activities, system events, and platform usage patterns.

ADVANCED GRADEBOOK FEATURES

Progress: 🟢🟢🟢🟢⬛⬛⬛⬛⬛⬛ (40%)

Target: A comprehensive grading system that enables instructors to grade assignments, quizzes, and discussions with flexible grading schemes (number-based and scale-based), rubric-based grading, advanced gradebook reporting with category totals and column collapsing, real-time grade calculations, and course completion tracking.

Current State: The gradebook structure and categories are complete, enabling instructors to organize grades hierarchically. The grading structure is defined for both number-based grading (numeric scores) and scale-based grading (letter grades or custom scales). The feedback system structure is in place. However, the grading mechanism is not yet implemented—instructors cannot currently grade assignments, quizzes, or discussions. The actual grading workflow where instructors can assign grades, provide feedback, and update student submissions is not functional.

Remaining Work: The grading mechanism must be implemented to enable instructors to grade assignments, quizzes, and discussions. Grader report category totals need to be implemented to show aggregated scores by category. Grader report column collapsing needs to be implemented to improve gradebook usability. Rubric-based grading system needs to be implemented to support detailed assessment criteria. Enhanced gradebook analytics and reporting need to be implemented to provide insights into student performance. Course completion and progress tracking need to be implemented to monitor student progress through courses.

SYSTEM ADMINISTRATION AND OPERATIONS

Progress: 🟢🟢🟢🟢🟢⬛⬛⬛⬛⬛ (50%)

Target: A comprehensive system administration and operations suite that enables automated task scheduling, media management, backup and restore capabilities, data export and import functionality, and administrative task automation.

Current State: Database migration management is complete in both admin panel and CLI mode, allowing administrators to check migration status and apply pending migrations. Database backup functionality is implemented through CLI mode, enabling administrators to create database dumps. However, a cron job system for automated tasks is not yet implemented. Media management and cleanup of unlinked media are not yet implemented. Backup and restore functionality for activity modules and courses are not yet implemented. Export to ZIP format for activity modules and courses is not yet implemented. Import functionality from ZIP, Moodle, and Canvas IMSCC formats is not yet implemented. Admin tasks runner is not yet implemented.

Remaining Work: A cron job system needs to be implemented to enable automated tasks such as scheduled backups, media cleanup, and other maintenance operations. Media management and cleanup of unlinked media need to be implemented to maintain storage efficiency. Backup and restore functionality for activity modules and courses need to be implemented to enable data portability. Export to ZIP format for activity modules and courses needs to be implemented. Import functionality from ZIP, Moodle, and Canvas IMSCC formats needs to be implemented to support data migration. Admin tasks runner needs to be implemented to enable automated administrative operations.

INTEGRATION AND API DEVELOPMENT

Progress: 🟢🟢⬛⬛⬛⬛⬛⬛⬛⬛ (20%)

Target: A comprehensive API-first architecture with public API access, OpenAPI documentation, LTI support, API key management, webhook system, and global search functionality that enables external integrations and custom development.

Current State: An internal API exists for internal use within the Paideia application, enabling the frontend to communicate with the backend. However, a public API is not yet implemented—external systems cannot currently access Paideia's functionality through API calls. OpenAPI documentation does not exist yet, so there is no documentation for API endpoints. LTI (Learning Tools Interoperability) support is not yet implemented. User and admin API key management are not yet implemented. Webhook system for external integrations is not yet implemented. Global search functionality is not yet implemented.

Remaining Work: A public API must be implemented to enable external systems to access Paideia's functionality programmatically. OpenAPI documentation must be created to document all API endpoints and enable developers to integrate with Paideia. LTI support must be implemented to enable integration with external educational tools. User and admin API key management must be implemented to enable secure API access. Webhook system must be implemented to enable external systems to receive notifications from Paideia. Global search functionality must be implemented to enable users to search across courses, modules, and content.

USER EXPERIENCE ENHANCEMENTS

Progress: 🟢🟢🟢🟢🟢🟢🟢⬛⬛⬛ (70%)

Target: A comprehensive user experience enhancement suite that provides responsive UI design optimization, quick command palette, breadcrumb navigation, global CSS and theme settings, course roadmap visualization, and shortcode system for content embedding.

Current State: Basic responsive design considerations are in place through Mantine UI components, which include built-in responsive design features. However, dedicated responsive UI design optimization has not yet been implemented—the platform has not been specifically optimized for mobile devices. Command K (quick command palette) is not yet implemented. Breadcrumb navigation is not yet implemented. Global CSS and theme settings are not yet implemented. Course roadmap visualization is not yet implemented. Shortcode system for content embedding is not yet implemented.

Remaining Work: Responsive UI design optimization must be implemented to ensure the platform works seamlessly across mobile devices, tablets, and desktop computers. Command K (quick command palette) must be implemented to enable users to quickly access features and navigate the platform. Breadcrumb navigation must be implemented to improve navigation and user orientation. Global CSS and theme settings must be implemented to enable institutions to customize the platform's appearance. Course roadmap visualization must be implemented to help students understand course structure and progress. Shortcode system for content embedding must be implemented to enable rich content embedding within course materials.

CONTENT AND COURSE MANAGEMENT

Progress: 🟢🟢🟢🟢🟢🟢🟢🟢⬛⬛ (80%)

Target: Enhanced content and course management capabilities including course curriculum management, fully public module access, course meta links and cross-linking, self-paced course configuration, content linting system, and mention functionality in rich text editor.

Current State: Course creation and organization are fully functional, allowing instructors to create courses, organize them into sections, and structure content hierarchically. Course categories and structure are implemented, enabling institutions to organize courses into logical groupings. However, course curriculum management is not yet implemented. Fully public module access (read-only for all users) is not yet implemented. Course meta links and cross-linking are not yet implemented. Self-paced course configuration is not yet implemented. Content linting system (dead links, hard internal links) is not yet implemented. Mention functionality in rich text editor is not yet implemented.

Remaining Work: Course curriculum management must be implemented to enable institutions to structure course content and learning paths. Fully public module access must be implemented to enable modules to be shared publicly in read-only mode. Course meta links and cross-linking must be implemented to enable courses to reference each other. Self-paced course configuration must be implemented to enable courses to be configured for self-paced learning. Content linting system must be implemented to identify and report dead links and hard internal links. Mention functionality in rich text editor must be implemented to enable users to mention other users, courses, and activity modules within content.

ASSESSMENT AND LEARNING TOOLS

Progress: ●●●●▪▪▪▪▪▪ (40%)

Target: Advanced assessment and learning tools including quiz submission functionality, flashcard system, image annotation question type for quizzes, ability to submit notes as assignment submissions, and conditional quiz logic for adaptive assessments.

Current State: The quiz module is functional with configuration options, enabling instructors to create quizzes with various question types. The assignment module is functional with configuration options, and assignment submission structure exists. However, quiz submission functionality is not yet implemented—students cannot currently submit quiz responses. Flashcard system is not yet implemented. Image annotation question type for quizzes is not yet implemented. Submit notes as assignment submissions is not yet implemented. Conditional quiz logic is not yet implemented.

Remaining Work: Quiz submission functionality must be implemented to allow students to submit quiz responses. Flashcard system must be implemented to support spaced repetition learning. Image annotation question type for quizzes must be implemented to enable students to annotate images as quiz responses. Submit notes as assignment submissions must be implemented to enable students to submit notes as assignment submissions. Conditional quiz logic must be implemented to enable adaptive assessments where questions are shown based on previous answers.

ANALYTICS AND MONITORING

Progress: ●●●●▪▪▪▪▪▪ (40%)

Target: A comprehensive analytics and monitoring system that provides analytics integration with external systems, page load statistics and timing, concurrent user monitoring, email sent logging and template viewer, and events and logging system for system observability.

Current State: Analytics and monitoring features are not yet implemented. Analytics integration with external systems is not yet implemented—the analytics system has not been chosen yet. Page load statistics and timing are not yet implemented. Concurrent user monitoring is not yet implemented. Email sent logging and template viewer are not yet implemented. Events and logging system are not yet implemented.

Remaining Work: Analytics integration with external systems must be implemented to provide insights into platform usage and performance—the specific analytics system will be chosen during implementation. Page load statistics and timing must be implemented to monitor platform performance. Concurrent user monitoring must be implemented to track system load and capacity. Email sent logging and template viewer must be implemented to enable administrators to monitor email communications and view email templates. Events and logging system must be implemented to provide system observability and debugging capabilities.

SECURITY AND COMPLIANCE

Progress: ▪▪▪▪▪▪▪▪▪▪ (0%)

Target: Comprehensive security audit and hardening, security best practices implementation, and compliance documentation to ensure the platform meets institutional security requirements and regulatory compliance standards.

Current State: Security and compliance features are not yet implemented. Comprehensive security audit and hardening have not yet been performed. Security best practices implementation is not yet complete. Compliance documentation is not yet available.

Remaining Work: Comprehensive security audit and hardening must be performed to identify and address security vulnerabilities. Security best practices implementation must be completed to ensure the platform follows industry security standards. Compliance documentation must be created to document security measures and compliance with regulatory requirements such as FERPA, GDPR, and COPPA.

DRIVE FEATURE

Progress: ██████████████ (0%)

Target: The Drive feature is one of the main features planned for version 1.0, providing comprehensive file and media management capabilities similar to Microsoft OneDrive. This feature will enable users to upload files and media, control access permissions, and share content with others. Users will be able to upload files and media, control access just like they control in Microsoft OneDrive, share public links or private links to other people, and share access based on specific user accounts.

Current State: The Drive feature is not yet implemented. No work has been done on this feature yet. Users cannot currently upload files and media through a drive interface. Access control management for files is not yet implemented. Public and private link sharing for files and media are not yet implemented. User-based access sharing is not yet implemented. Folder organization and file management are not yet implemented. Integration with course materials and activity modules is not yet implemented.

Remaining Work: The Drive feature must be implemented to provide comprehensive file and media management capabilities. File and media upload functionality must be implemented to enable users to upload files and media. Access control management must be implemented to enable users to control access permissions similar to Microsoft OneDrive. Public link sharing must be implemented to enable users to share files and media with public links. Private link sharing must be implemented to enable users to share files and media with private links and controlled access. User-based access sharing must be implemented to enable users to grant access to specific user accounts. Folder organization and file management must be implemented to enable users to organize files and media. Integration with course materials and activity modules must be implemented to enable files and media to be used within courses and activities.

This feature will provide institutions with a centralized file management system integrated directly into the learning management platform, eliminating the need for external file storage solutions.

ADDITIONAL FEATURES

Progress: 🟢🟢🟢🟢🟢███████ (50%)

Target: Additional features including calendar system, scheduled course creation, portfolio functionality, certificate generation, module creator acknowledgement, and module rating system to enhance platform capabilities and user engagement.

Current State: Additional features are not yet implemented. Calendar system is not yet implemented. Scheduled course creation is not yet implemented. Portfolio functionality is not yet implemented. Certificate generation is not yet implemented. Module creator acknowledgement is not yet implemented. Module rating system (helpful/not helpful) is not yet implemented.

Remaining Work: Calendar system must be implemented to enable users to view and manage course schedules and events. Scheduled course creation must be implemented to enable courses to be created automatically on scheduled dates. Portfolio functionality must be implemented to enable students to showcase their work. Certificate generation must be implemented to enable institutions to generate certificates for course completion. Module creator acknowledgement must be implemented to recognize module creators. Module rating system must be implemented to enable users to rate modules as helpful or not helpful.

LONG-TERM VISION

Version 2.0 and Beyond

Paideia's long-term vision encompasses advanced integrations, AI capabilities, and enhanced collaboration features:

ENTERPRISE INTEGRATIONS

Progress: �*████████████* (0%)

Target: Comprehensive enterprise integrations including Microsoft built-in integration (Teams, OneDrive, Office 365), Obsidian integration for note-taking workflows, and Anki integration for spaced repetition learning to enable seamless connectivity with institutional infrastructure and learning tools.

Current State: Enterprise integrations are not yet implemented. Microsoft built-in integration (Teams, OneDrive, Office 365) is not yet implemented. Obsidian integration for note-taking workflows is not yet implemented. Anki integration for spaced repetition learning is not yet implemented.

Remaining Work: Microsoft built-in integration must be implemented to enable seamless connectivity with Microsoft Teams, OneDrive, and Office 365, providing authentication, file storage, and communication capabilities. Obsidian integration must be implemented to enable users to integrate their note-taking workflows with Obsidian. Anki integration must be implemented to enable users to integrate spaced repetition learning with Anki.

AI AND AUTOMATION

Progress: ▄████████████ (0%)

Target: AI capabilities using Model Context Protocol (MCP), AI-enhanced content creation and assessment, and intelligent automation for administrative tasks to enhance platform capabilities and reduce administrative burden.

Current State: AI and automation features are not yet implemented. AI capabilities using Model Context Protocol (MCP) are not yet implemented. AI-enhanced content creation and assessment are not yet implemented. Intelligent automation for administrative tasks is not yet implemented.

Remaining Work: AI capabilities using Model Context Protocol (MCP) must be implemented to enable AI-enhanced features throughout the platform. AI-enhanced content creation and assessment must be implemented to enable AI-assisted content creation and assessment. Intelligent automation for administrative tasks must be implemented to reduce administrative burden and improve efficiency.

ADVANCED PLATFORM CAPABILITIES

Progress: ▄████████████ (0%)

Target: Advanced platform capabilities including headless CMS mode for flexible deployments, networking and social learning features, internationalization (i18n) at global and course levels, and online meeting integration to enhance platform flexibility and global reach.

Current State: Advanced platform capabilities are not yet implemented. Headless CMS mode for flexible deployments is not yet implemented. Networking and social learning features are not yet implemented. Internationalization (i18n) at global and course levels is not yet implemented. Online meeting integration is not yet implemented.

Remaining Work: Headless CMS mode must be implemented to enable flexible deployments and custom frontend implementations. Networking and social learning features must be implemented to enable social interactions and networking within the platform. Internationalization (i18n) at global and course levels must be implemented to enable the platform to support multiple languages and regional requirements. Online meeting integration must be implemented to enable integration with online meeting platforms.

ENHANCED ASSESSMENT FEATURES

Progress: ▰▰▰▰▰▰▰▰▰▰ (0%)

Target: Enhanced assessment features including conditional quiz logic and adaptive assessments, and advanced question types and assessment tools to provide sophisticated assessment capabilities.

Current State: Enhanced assessment features are not yet implemented. Conditional quiz logic and adaptive assessments are not yet implemented. Advanced question types and assessment tools are not yet implemented.

Remaining Work: Conditional quiz logic and adaptive assessments must be implemented to enable assessments where questions are shown based on previous answers. Advanced question types and assessment tools must be implemented to provide sophisticated assessment capabilities beyond basic quiz questions.

USER EXPERIENCE AND ONBOARDING

Progress: ▰▰▰▰▰▰▰▰▰▰ (0%)

Target: User experience and onboarding features including interactive user tours and onboarding, auto-grouping functionality for course participants, and course-level impersonation for instructors to enhance user experience and reduce onboarding time.

Current State: User experience and onboarding features are not yet implemented. Interactive user tours and onboarding are not yet implemented. Auto-grouping functionality for course participants is not yet implemented. Course-level impersonation for instructors is not yet implemented.

Remaining Work: Interactive user tours and onboarding must be implemented to help new users learn the platform and reduce onboarding time. Auto-grouping functionality for course participants must be implemented to automatically organize students into groups based on criteria. Course-level impersonation for instructors must be implemented to enable instructors to view courses from a student's perspective.

VERSION CONTROL AND COLLABORATION

Progress: ▰▰▰▰▰▰▰▰▰▰ (0%)

Target: A comprehensive PostgreSQL-native version control system for course materials and activity modules that enables branch-based course development workflows, supports instructor and instructional designer collaboration models, and provides complete version control capabilities without requiring external Git integration.

Current State: Version control and collaboration features are not yet implemented. The system does not currently support version control for course materials or activity modules. Branch-based workflows are not available. Collaboration models between instructors and instructional designers are not yet supported.

Remaining Work: A complete PostgreSQL-native version control system needs to be implemented for course materials. Branch-based course development workflows must be implemented to enable instructors to branch off from main course content and merge back changes. Instructor and instructional designer collaboration models need to be implemented to support collaborative course development.

Activity module version control and branching need to be implemented to support collaborative module development.

PROGRAM AND CURRICULUM MANAGEMENT

Progress: ▒▒▒▒▒▒▒▒▒▒▒ (0%)

Target: A comprehensive program and curriculum management system that enables users to join educational organizations as students in specific programs (e.g., Bachelor of Business Administration), administrators and content managers to set curricula for different programs with defined timelines (e.g., four-year programs), users to follow curriculum maps and build their own curriculum maps, and users to enroll in courses directly within the system without requiring external third-party software or systems.

Current State: UI mockups have been created for the dashboard, curriculum map, and course enrollment page, demonstrating how these features would look and function within Paideia. However, the actual program and curriculum management features are not yet implemented. Users cannot currently join educational organizations as students in specific programs. Administrators and content managers cannot set curricula for different programs. Curriculum maps are not yet implemented. Users cannot follow curriculum maps or build their own curriculum maps. Direct course enrollment by users is not yet implemented—course enrollment is currently managed by administrators and instructors.

The development team is currently uncertain about whether this feature should be built into Paideia or remain a capability that should be handled by external applications. This decision will be informed by community feedback and user input, as the team seeks to understand whether institutions prefer integrated program and curriculum management within the LMS or prefer to use specialized external systems for these functions.

Remaining Work: The development team will gather community feedback and user input to determine whether program and curriculum management should be built into Paideia or remain a capability handled by external applications. If the decision is made to build this feature into Paideia, a program management system must be implemented to enable users to join educational organizations as students in specific programs (e.g., Bachelor of Business Administration, Master of Science, etc.). A curriculum management system must be implemented to enable administrators and content managers to set curricula for different programs with defined timelines (e.g., four-year undergraduate programs, two-year graduate programs). Curriculum maps must be implemented to enable users to visualize and follow their program requirements. User-driven curriculum map building must be implemented to enable users to build their own curriculum maps based on program requirements. Direct course enrollment by users must be implemented to enable users to enroll in courses directly within the system without requiring external third-party software or systems, distinguishing Paideia from Moodle and Canvas where course enrollment is typically handled by external student information systems.

This feature represents a significant departure from traditional LMS platforms like Moodle and Canvas, where course enrollment is typically handled by external student information systems (SIS) or third-party software. By building program and curriculum management directly into Paideia, institutions could manage the entire student lifecycle—from program enrollment to course selection to graduation—within a single, integrated platform. This approach would reduce complexity, eliminate the need for external enrollment systems, and provide institutions with complete control over their educational programs and student pathways. However, the development team recognizes that some institutions may prefer to use specialized external systems for program and curriculum management, and the final decision will be guided by community needs and preferences.

Strategic Vision

Paideia envisions a future where learning management systems are effortlessly deployable through single-binary architecture, version-controlled at the course and module level, AI-enhanced without compromising human-centered learning, integration-ready rather than monolithically complex, and cost-effective for institutions of all sizes.

Our long-term sustainability model centers on maintaining an open source foundation where the core platform remains free and open source. Revenue will come through professional services including hosting, support, and custom integrations. Feature development will be community-driven, guided by educator and administrator needs. Platform stability remains a commitment, with non-breaking updates and zero data loss as fundamental guarantees.

FUTURE ROADMAP

Feature Completion State

Paideia will reach a "feature complete" state where no additional core features are added to the platform. This milestone represents a significant achievement in the platform's development lifecycle, marking the transition from active feature development to platform maturity and stability.

Beyond this point, the development focus will shift fundamentally. The primary emphasis will be on bug fixes and performance improvements, ensuring that existing features work flawlessly and efficiently. Rather than expanding the core platform with new built-in functionality, new capabilities will be delivered through integrations with external systems and services. This approach maintains the platform's core simplicity while enabling institutions to extend functionality according to their specific needs.

Platform stability will become the primary concern, with development efforts concentrated on ensuring reliability, security, and performance at scale. The community-driven enhancement process will continue to guide improvements, but the focus will be on refining existing features rather than introducing new ones. This approach ensures that Paideia remains maintainable, reliable, and focused on its core mission of providing a simple, effective learning management system.

Integration Ecosystem

Rather than building every possible feature internally, Paideia will establish a comprehensive integration ecosystem that enables institutions to extend the platform's capabilities according to their specific requirements. This approach maintains the platform's core simplicity while providing flexibility for diverse institutional needs.

The integration ecosystem will provide robust APIs for third-party developers, enabling them to build custom integrations and extensions that seamlessly connect with Paideia's core functionality. These APIs will be well-documented, stable, and designed to support a wide range of integration scenarios. The platform will support standard educational technology protocols, including LTI (Learning Tools Interoper-ability), enabling seamless integration with existing educational tools and services.

Custom integration development will be enabled through comprehensive documentation, developer resources, and support channels. Institutions and developers will be able to create custom integrations that meet their specific needs, whether for student information systems, authentication providers, content repositories, or specialized educational tools. The platform will maintain a marketplace of community extensions, where developers can share their integrations and institutions can discover and implement solutions that meet their requirements.

This integration-first approach ensures that Paideia remains focused on its core mission while enabling institutions to build comprehensive educational technology ecosystems tailored to their specific needs.

Community Development and Engagement

Paideia's success depends on building a vibrant, engaged community of educators, administrators, developers, and educational technology enthusiasts. The project will actively foster community development and engagement through multiple channels and initiatives.

GitHub will serve as the primary platform for community collaboration, where developers can contribute code, report issues, participate in discussions, and collaborate on feature development. The project will

maintain transparent development processes, with all code changes, discussions, and decisions visible to the community. Regular communication about project direction, feature priorities, and development progress will keep the community informed and engaged.

YouTube will be used to create extensive educational content, including explainer videos that teach users how to use Paideia, tutorials for administrators setting up and managing the platform, and technical content for developers interested in contributing or building integrations. These videos will make the platform more accessible to non-technical users and provide comprehensive guidance for all aspects of using and managing Paideia.

Social media platforms, particularly X (formerly Twitter), will be used to share updates, announcements, and engage with the community. Regular posts about new features, development progress, community highlights, and educational technology insights will keep the community informed and engaged. Social media will also serve as a platform for gathering feedback, answering questions, and building relationships with the broader educational technology community.

Documentation will be continuously improved and expanded as features become more stable. The documentation site at https://docs.paideialms.com will be maintained as the authoritative source for all platform documentation, including user guides, administrator manuals, developer resources, and API documentation. As features mature and stabilize, comprehensive documentation will be created to ensure that users, administrators, and developers have the resources they need to effectively use, manage, and extend the platform.

This multi-channel approach to community development ensures that Paideia remains accessible, well-documented, and supported by an active, engaged community of users and contributors.

CONCLUSION

Paideia LMS represents more than just another learning management system—it embodies a fundamental reimagining of what a learning management system can be when built from the ground up with modern principles, open source values, and educational excellence in mind. With the release of v0.5, we have demonstrated that rapid, high-quality development is possible while maintaining our core principles of being free forever, technically superior, and operationally efficient.

In just over a month since development began on September 29, 2025, Paideia has achieved a demonstratable platform that addresses the fundamental limitations of existing LMS platforms. The first launch of version 0.5 on November 3, 2025 demonstrates significant progress across core platform infrastructure, dual-mode binary architecture with CLI capabilities, comprehensive user and course management, functional learning activity modules, content management capabilities, and established gradebook structure. The demo online version at demo.paideialms.com with sandbox mode provides institutions with immediate access to evaluate Paideia's capabilities.

Our commitment to free software under AGPL-3.0 ensures that quality educational technology remains accessible to institutions regardless of their financial resources. Through careful technical architecture leveraging modern technologies—TypeScript, Bun, React, PostgreSQL, and S3-compatible storage—and community-driven development, Paideia is establishing itself as the premier alternative to legacy LMS platforms. The platform's value proposition is compelling: 50-80% cost reduction compared to alternatives, deployment in minutes, minimal infrastructure requirements, complete data ownership, and horizontal scalability.

Paideia's operational principles guide the project's direction: open source and free forever, architectural simplicity, transparency and openness, and community-driven development. Design principles emphasize balanced experience design across teaching, learning, and administration; balanced feature design balancing minimalism, battery-included features, and flexibility; and a modern technology stack built on proven, sector-leading technologies. These principles ensure that Paideia remains focused on its core mission while enabling institutions to build comprehensive educational technology ecosystems tailored to their specific needs.

The short-term vision focuses on completing version 1.0 with core activity modules completion, advanced gradebook features, logging and event handling, system administration capabilities, integration and API development, user experience enhancements, and the Drive feature. The long-term vision encompasses enterprise integrations, AI and automation capabilities, advanced platform features, and version control and collaboration systems. Community development and engagement will be fostered through GitHub, YouTube, social media platforms like X, and comprehensive documentation as features become stable.

The future of educational technology is efficient, collaborative, and AI-enhanced. With v0.5, Paideia has taken significant steps toward leading this transformation while maintaining the human-centered approach that makes education meaningful. As we move forward, Paideia will continue to evolve through community-driven development, maintaining its commitment to being free forever, technically superior, and operationally efficient. The platform's journey from concept to v0.5 demonstrates that modern educational technology can be both powerful and accessible, complex in capability yet simple in operation, and innovative in design while remaining grounded in the fundamental needs of educators, students, and administrators.

ACKNOWLEDGEMENTS

Paideia LMS would not have been possible without the exceptional technologies, frameworks, and open source projects that form its foundation. We extend our sincere gratitude to the developers and communities behind these technologies for their contributions to the open source ecosystem.

Technologies and Frameworks

Paideia is built upon a modern technology stack that includes:

- **TypeScript** for comprehensive type safety
- **Bun** for runtime and bundling capabilities
- **React** for user interface development
- **React Router v7** for server-side rendering and routing
- **Elysia** for high-performance web framework capabilities
- **Payload CMS** for content management system functionality
- **Drizzle ORM** for type-safe database operations
- **PostgreSQL** for robust and scalable data storage
- **S3-compatible storage** for media and file management
- **Mantine** for comprehensive UI components
- **TipTap** for rich text editing
- **Excalidraw** for whiteboard functionality
- **Vite** for development server capabilities

These technologies represent the best of modern web development, providing the performance, reliability, and developer experience that enable Paideia to deliver on its promises.

We are grateful to the open source communities that maintain and improve these technologies, making it possible for projects like Paideia to leverage cutting-edge capabilities without the burden of building everything from scratch. The open source ecosystem's commitment to innovation, quality, and accessibility directly enables Paideia's mission to provide free, accessible educational technology.

Systems Studied

In developing Paideia, we have studied and learned from existing learning management systems, particularly Moodle and Canvas. These platforms have served the educational technology community for many years and have provided valuable insights into the challenges and opportunities in the LMS space. Our analysis of Moodle's open source approach, plugin architecture, and community-driven development has informed our understanding of what works well and what could be improved. Similarly, our study of Canvas's user experience design, modern interface, and institutional adoption patterns has contributed to our vision for Paideia's design principles and user experience goals.

While Paideia takes a different architectural and operational approach, we acknowledge the significant contributions that Moodle and Canvas have made to educational technology. These platforms have demonstrated the importance of learning management systems in modern education and have helped establish the expectations and requirements that institutions have for LMS platforms. Our goal is not to replicate these systems, but to learn from their successes and challenges while building a modern alternative that addresses fundamental limitations in cost, complexity, and technical debt.

Open Source Community

Finally, we acknowledge the broader open source community that makes projects like Paideia possible. The open source movement's commitment to free software, collaborative development, and knowledge sharing creates an environment where innovative educational technology can flourish. We are grateful to be part of this community and are committed to contributing back through Paideia's open source development, documentation, and community engagement.

AUTHOR INFORMATION

- **Yeung Man Lung Ken**

  Email: `yomaru@paideialms.com`

  𝕏 https://x.com/yomaru_1999

  https://github.com/HananoshikaYomaru

  https://www.youtube.com/@yomaru_1999

PAIDEIA LMS: A MODERN LEARNING MANAGEMENT SYSTEM

Introducing Paideia LMS: A modern, AI-native alternative to Moodle and Canvas designed to lower administration and hosting costs.

11/03/2025

KEN YEUNG