Vol. 7, No. 4, December 2025

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Multi-Tier Architecture Design for Scalable and Effective Non-Formal Learning: A Redesign of Serat Kartini Women's School LMS

Primavieri Rhesa Ardana[1], Gustina Alfa Trisnapradika[2]

[1,2]Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia

Email: 111202214557@mhs.dinus.ac.id[1], gustina.alfa@dsn.dinus.ac.id[2]

Received: October 21, 2025

Revised: Nov 1, 2025

Accepted: Nov 12, 2025

Published: Dec 13, 2025

Corresponding Author:

Author Name*:

Gustina Alfa Trisnapradika

Email*:

gustina.alfa@dsn.dinus.ac.id

DOI:

10.63158/journalisi.v7i4.1341

Abstract. Non-formal education plays a vital role in empowering women in rural areas of Central Java, Indonesia. However, the existing Learning Management System (LMS) of Woman School Serat Kartini, built on a monolithic Laravel architecture, suffers from significant performance degradation and scalability limitations under growing user loads and shared hosting constraints. This leads to high latency, frequent session interruptions, and reduced participation, ultimately undermining learning effectiveness. This study redesigns the LMS using a multi-tier application architecture through the Design Science Research (DSR) methodology. The proposed blueprint separates the system into four independent tiers: Presentation (Next.js for users, React.js for administrators), Logic (Express.js for API Layer), Cache (Redis with cache-aside strategy), and Data (MySQL). The design artifacts include detailed architecture diagrams, ERD, use case, and sequence diagrams. Conceptual evaluation demonstrates that the multi-tier approach enhances modularity, reduces latency, supports horizontal scalability, and improves resource efficiency , ensuring reliable access for women learners with limited digital literacy and unstable internet connectivity. The redesigned LMS conceptually strengthens learning accessibility, engagement, and program sustainability in resource-constrained non-formal education contexts. This research is limited to the conceptual design phase without implementation or empirical testing.

Keywords: Multi-Tier Architecture, Design Science Research, Non-Formal Education, Learning Effectiveness, Learning Management System

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

Nonformal education plays a crucial role in empowering women, particularly in regions with limited access to formal education systems [1]. In this context, Learning Management Systems (LMS) hold significant potential to facilitate learning and skill development, such as enhancing critical thinking among learners [2]. Initiatives like the Woman School Serat Kartini, operating under the auspices of DP3AP2KB in Central Java, exemplify key efforts to bolster women's capabilities through nonformal pathways by delivering structured educational content and tracking progress [3]. Thus, the availability of an effective, accessible, and scalable LMS is paramount to ensuring the sustainability and broader reach of such programs.

Existing research on LMS implementations predominantly features monolithic architectures, where presentation, logic, and data tiers are tightly integrated into a single codebase and deployment unit, facilitating straightforward initial development and maintenance for smaller-scale applications [4], [5]. Among these, studies highlight Moodle and similar open-source platforms as among the best-performing monolithic solutions due to their modular plugins, community-driven enhancements, and cost-effectiveness, achieving up to 80% resource efficiency in controlled educational environments with moderate user loads . Research on proprietary systems, such as those transitioning from on-premises to cloud-based hosting, further underscores their strengths in seamless multimedia integration and reduced administrative overhead by 30-50%, as demonstrated in collaborative e-learning projects [6]. Additionally, performance evaluations of monolithic LMS in junior high school settings reveal high user satisfaction for basic tracking and reporting features, with response times under 500 ms for standard operations [7] . Other investigations into web-based service applications emphasize the simplicity of monolithic designs for rapid prototyping, enabling quick feature additions without complex inter-service coordination [5]. Notably, frameworks like Laravel have been identified as exemplary choices within monolithic architectures for LMS development, offering robust MVC (Model-View-Controller) structures, built-in security features, and ease of integration with databases like MySQL, which contribute to efficient handling of user authentication, content management, and scalability in initial deployments. This approach is particularly relevant as the current LMS for Woman School Serat Kartini employs Laravel, demonstrating its effectiveness in supporting nonformal

education programs with limited initial resources while providing a solid foundation for core functionalities such as course delivery and progress tracking.

However, these monolithic approaches, including Laravel-based systems, exhibit significant limitations, particularly in scalability and flexibility as user bases and features expand [5]. Comparative analyses indicate that monolithic systems suffer from performance degradation during traffic spikes, with full redeployments required for minor updates, leading to increased downtime and maintenance costs—often exceeding 20% of development time [8]. Refactoring studies from monolithic to microservices architectures, using decomposition and strangler patterns, confirm these issues, showing initial monolithic setups falter under high concurrency, with query latencies rising by up to 2 seconds and overall throughput dropping by 40% in overloaded scenarios [8]. Industry insights on microservices migration reveal that while monolithic designs excel in simplicity, they inadequately handle distributed geographical demands or resource-constrained environments like shared hosting, where tight component coupling hinders independent optimization and exacerbates bottlenecks in data access [9]. Moreover, in e-learning contexts, monolithic LMS often overlook advanced caching integration, resulting in inefficient relational database queries that amplify latencies for users with low digital literacy or slow internet [10], [11]. These performance deficits directly translate into a negative user experience and reduced learning effectiveness, thereby compromising the overall goals of the non-formal education program. This vulnerability to performance issues is acutely manifested in resource-constrained contexts**.**

In the context of nonformal institutions like Woman School Serat Kartini, which rely on limited resources under DP3AP2KB in Central Java, LMS scalability emerges as a critical challenge due to dependence on affordable shared hosting [12]. Such hosting, costing as low as IDR 50,000 monthly from local providers, partitions server resources (e.g., CPU, RAM, bandwidth) among multiple users, making it prone to overload during surges, such as hundreds of participants from various Central Java cities accessing materials or online classes simultaneously. Crucially, observations indicate that under peak loads, the current monolithic system exhibits query latencies exceeding 500 milliseconds, resulting in session interruptions and user frustration. This technical friction directly impacts learning effectiveness; program administrators report a significant module drop-off rate, with many participants failing to complete course sequences due to inaccessible materials or

session timeouts**.** These issues undermine consistent program participation and hinder the achievement of the program's core empowerment goals. Conventional monolithic architectures, such as the Laravel-based system currently in use, worsen this by integrating all components into one deployment, causing holistic performance dips and downtime that disrupt empowerment programs, where uninterrupted access is essential for participants with flexible schedules and varying digital literacy. These constraints also bar adoption of costly dedicated servers or premium cloud services, demanding solutions that manage loads efficiently with minimal investment, while mitigating external factors like rural internet delays that heighten system latency.

Multi-tier architectures mitigate these drawbacks via separation of concerns, promoting horizontal scalability at low cost without major hardware overhauls [13]. For instance, decoupling the Presentation Tier (Next.js for user interfaces, React.js for admin panels), Logic Tier (Express.js API), Data Tier (MySQL), and Cache Tier (Redis) enables independent component scaling: Redis can cache transient data to slash database queries by 60-80%, with empirical tests yielding 3.3-fold gains in access speed, thus preserving shared hosting viability, sustaining responsiveness, and stabilizing performance amid user peaks [11], [12]. This modular deployment further curtails downtime risks—vital for understaffed nonformal schools—and eases upkeep for part-time IT teams [14]. A cache-aside Redis strategy bolsters efficiency in constrained setups by in-memory storing frequent data (e.g., login sessions, class materials), easing MySQL bottlenecks prevalent on shared platforms [12].

A few researchers have focused on monolithic LMS designs for general educational contexts, prioritizing initial simplicity and cost savings. There have been limited studies concerned with multi-tier redesigns tailored to resource-limited nonformal institutions in developing regions, particularly integrating modern stacks like Next.js, Express.js, React.js, MySQL, and Redis to optimize performance for women's empowerment amid shared hosting constraints. Therefore, this research intends to develop a comprehensive redesign for a scalable LMS based on Multi-Tier Architecture for Woman School Serat Kartini. The objectives of this research are to perform in-depth needs analysis, delineate detailed system architecture, optimize database schemas, and produce system diagrams (e.g., activity, sequence, and component). Through these design artifacts, the study conceptually validates the proposed design's feasibility and efficacy, encompassing

theoretical assessments of functional fulfillment, usability support, and superior performance/efficiency over monolithic baselines, which are essential technical prerequisites for achieving the program's learning effectiveness objectives. This blueprint aims to ensure theoretical operability, user intuitiveness, and substantive performance uplift potential for future realization.

This research contributes theoretically by advancing Multi-Tier Architecture literature in nonformal LMS, emphasizing performance, scalability, and efficiency via contemporary tech integrations, and practically by furnishing a replicable model for responsive, scalable platforms that guarantee accessibility, thereby directly enhancing women's educational access and effectiveness in Central Java. The scope centers on conceptual and detailed design of the scalable LMS redesign for Woman School Serat Kartini, halting at comprehensive design without code implementation or empirical testing. Reviewed technologies include Next.js (user interfaces), Express.js (APIs), React.js (admin interfaces), and Redis cache-aside strategies. Core features encompass user-side class/material modules and full admin management modules.

2. METHODOLOGY

This chapter details the methodological approach adopted for the redesign of the Learning Management System (LMS) at Woman School Serat Kartini. The overall research process follows the systematic framework of Design Science Research (DSR), emphasizing the creation of a technological artifact, the multi-tier architecture blueprint to address the pressing practical problems identified in the current system.

This research employs the Design Science Research (DSR) methodology, following the process model proposed by Peffers et al. [15], to redesign the Learning Management System (LMS) of Woman School Serat Kartini. Woman School Serat Kartini is a non-formal educational institution dedicated to women's empowerment in rural Central Java, Indonesia. DSR is selected because it provides a rigorous and systematic framework for creating and evaluating innovative IT artifacts that solve identified organizational problems while contributing to both scientific knowledge and practical impact.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Within the DSR framework, the development of the primary artifact (a multi-tier architecture blueprint) is conducted using an iterative prototyping approach aligned with agile principles. This iterative method enables continuous refinement of the design based on ongoing analysis of functional requirements, performance objectives, and feedback loops derived from the institution's real operational constraints (shared hosting, limited technical staff, and users with varying digital literacy). The combination of DSR as the overarching research paradigm and agile-oriented iterative prototyping ensures that the resulting blueprint is theoretically sound, practically feasible, and directly responsive to the needs of non-formal women's education programs.

This methodological choice is particularly appropriate given empirical evidence that distributed architectures, such as multi-tier designs, deliver superior reliability, fault tolerance, and horizontal scalability compared to traditional monolithic systems [16].. The current monolithic LMS exhibits latency exceeding 500 ms under loads of approximately 500 concurrent users [17]. Such latency leads to session interruptions that negatively affect learning engagement and program completion rates among female participants. By grounding the redesign process in DSR and iterative prototyping, this study aims to produce a scalable, efficient, and inclusive LMS artifact that conceptually enhances accessibility and learning effectiveness in resource-constrained non-formal education settings.

2.1. Design Science Research Framework

Following the systematic process outlined by Peffers *et al.* [15], the research is organized into six conceptual stages: problem identification, objectives determination, artifact design and development, demonstration, evaluation, and communication of results. The workflow is illustrated in Fig. 1, which has been adapted to reflect the specific context of resource-constrained deployment on shared hosting platforms. The study is limited to the first three stages, problem identification, objective determining, and artifact design, while demonstration and evaluation are conducted conceptually through architectural modeling, sequence diagrams, and performance projections. Full implementation and empirical testing are excluded due to institutional constraints, including limited technical personnel and reliance on low-cost shared hosting infrastructure. Therefore, this study focuses on delivering a robust and validated conceptual blueprint, which serves as the

essential prerequisite for the subsequent DSR cycle involving empirical implementation and effectiveness testing.

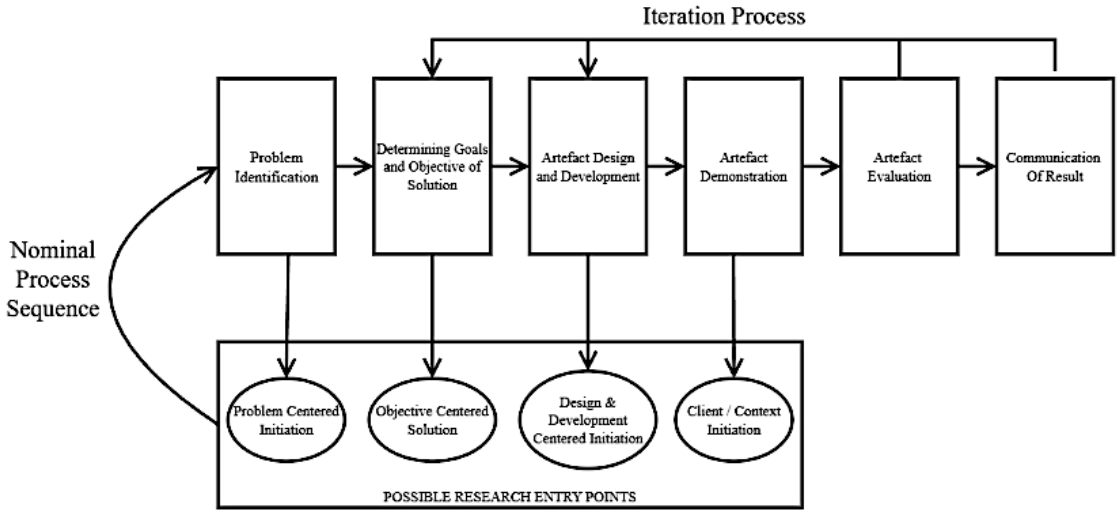

Figure 1. Design Science Research Methodology Process [15]

2.2. Research Stage Based On DSR

The research followed the systematic stages of the DSR framework (Peffers et al.) sequentially. The artifact design and development phase is guided by an Iterative Development Approach (Prototyping) to refine the multi-tier architectural blueprint based on continuous objective determination**,** which is implicit in the core design cycle. The application of each stage was tailored to the objective of redesigning the LMS architecture. The following sub-sections detail how the research progressed through the formal DSR steps, from identifying the practical deficiencies of the existing system to conceptually evaluating the resultant multi-tier design artifact**.**

2.3.1 Problem Identification

The current LMS deployed at Woman School Serat Kartini faces significant performance and scalability constraints, primarily stemming from its inflexible monolithic architecture. In-depth interviews with school administrators and system observations identified critical issues, including high latency (exceeding 500 milliseconds during simultaneous usage by over 500 users), which negatively impacts the user experience. Furthermore, the limited scalability complicates the integration of advanced features and hinders the school's strategic plan for program expansion into other regions. The inherent tight

coupling in the monolithic structure also makes system maintenance and the adoption of modern technologies challenging. These issues underscore the urgent need for a robust, multi-tier architectural solution capable of supporting high concurrency and future growt

Table 1. Problem Identification

Issue	Impact	Affected Component
High latency (>500 ms)	Degraded user experience, session interruptions	Presentation & Data Tier
Limited scalability	Blocked feature expansion (e.g., analytics)	Overall Architecture
Maintenance complexity	Elevated downtime and recovery time	Deployment & Logic Tier
No task segregation	Inefficient scaling and resource allocation	System Modularity
Peak load handling failure	Reduced productivity and program efficacy	Concurrent User Handling

2.3.2 Determining Goals and Solution

Based on the problems identified in Table 1, particularly the high latency causing session interruptions and the limited scalability inherent in the monolithic design, the primary objective is to develop a comprehensive multi-tier architecture blueprint designed to enhance the LMS's modularity and performance. Specifically, to address the lack of task segregation and maintenance complexity highlighted in the table, the design must achieve a clear separation of concerns across the Presentation, Logic, Cache, and Data Tiers, facilitating independent development and maintenance . A critical performance goal is to reduce system latency below 100 milliseconds and effectively support up to 500 concurrent users through the strategic integration of a dedicated Cache Tier. This architectural design is tailored to meet the specific functional requirements of the Serat Kartini school, including class management, material distribution, and comprehensive evaluation processes.

Furthermore, the design ensures operational resilience and adaptability to user growth, particularly in rural and low-connectivity areas of Central Java. By prioritizing horizontal scalability, modular updates, low-latency data retrieval via cache-aside patterns, and minimal hardware demands, the solution guarantees sustained accessibility and

inclusivity for female learners, even under variable network conditions and limited institutional IT support. These objectives collectively establish a foundation for a future-proof, cost-effective LMS that not only resolves current performance deficits but also supports strategic program expansion, pedagogical innovation, and long-term sustainability within the financial and technical realities of nonformal education providers.

Table 2. Goals and Solution

No	Objective	Target Metric	Justification
1	Response Time	< 100 ms (95th percentile)	Ensures rapid access to learning materials, critical for users with low digital literacy and unstable internet in rural nonformal education settings [7].
2	Scalability	Support 500+ concurrent users	Accommodates projected peak enrollment across multiple districts in Central Java, preventing system overload during mass training sessions [3].
3	Modularity	Four-tier separation of concerns	Enables independent development and deployment of Presentation, Logic, Cache, and Data tiers, reducing maintenance complexity in monolithic systems [4], [8].
4	Deployment Cost	< Rp 50.000/month (~$3 USD)	Compatible with low-cost shared hosting used by nonprofit and nonformal institutions, ensuring financial sustainability without cloud migration [6].
5	Maintainability & Performance	Cache-aside strategy with Redis; reduce DB load by ≥70%	Leverages in-memory caching to accelerate relational data access and improve system efficiency under resource-constrained environments [11], [12]

2.3.3 Artifact Design and Development

This design phase utilizes an Iterative Development Approach (Prototyping) to refine the multi-tier architectural blueprint, aligning the artifact creation with the cyclical nature inherent in the DSR process of objective determination and solution refinement. Based on the goals and solutions outlined in Table 2, particularly the requirement for a four-tier separation of concerns to ensure modularity , the artifact developed in this phase is the detailed design of a four-tier application architecture. The Presentation Tier is

designed using Next.js for the public interface and React.js for the segregated administrative dashboard, ensuring responsiveness and SEO optimization. The Logic Tier is implemented with Express.js to function as a lightweight, centralized RESTful API, handling core business logic. Crucially, to fulfill the maintainability and performance targets mentioned in Table 2 , the Cache Tier utilizes Redis for in-memory storage of frequently accessed data, while the Data Tier relies on MySQL as the relational storage engine. This layered design enhances system modularity, scalability, stability, and efficiency.

Table 3. Technical Spesification and Strategy Per Tier

Tier	Technology	Core Responsibilities	Scalability Strategy
Presentation	Next.js (User), React.js (Admin)	UI rendering, SEO, responsive design	Horizontal (stateless instances)
Logic	Express.js	API gateway, business rules, validation	Independent deployment, micro-API updates
Cache	Redis	Session & content caching, latency reduction	Vertical (memory scaling), cache-aside pattern
Data	MySQL (seratkartini schema)	Persistent storage, relational integrity	Read replicas, query optimization, Column Indexing

As detailed in Table 3, which outlines the core responsibilities and specific scalability strategies for each layer, this tiered separation ensures that updates to any layer do not require full system redeployment, a critical advantage on shared hosting where downtime is costly and technical expertise is limited. The architecture is deliberately lightweight, with each tier optimized for minimal resource footprint, enabling reliable operation on entry-level shared servers while supporting up to 500 concurrent users and future expansion across multiple regions in Central Java. Detailed diagrams and specifications are deferred to Chapter 3 to maintain methodological clarity.

2.3.4 Artefact Demonstration

The demonstration phase showcases the viability and functionality of the multi-tier architectural design artifact. Since the research is confined to the conceptual and detailed design phases, the demonstration is performed through architectural visualization and functional modeling rather than coded implementation. The

demonstration focuses on illustrating how the proposed four-tier structure addresses the core problems of the legacy system. This is achieved by presenting the architectural specifications, which detail the four tiers, and mapping the system's operational flow using Use Case Diagrams. Crucially, the efficiency of the new system, particularly the role of the Cache Tier, is demonstrated using a Sequence Diagram that models the cache-aside strategy. Furthermore, mockup UIs are provided to visualize the user-centric output of the Presentation Tier. The resulting visualizations and models are presented in detail in Chapter 3**.**

2.3.5 Artefact Evaluation

Evaluation is conducted as an ex-ante conceptual assessment to determine the artifact's fit for the purpose prior to actual deployment. This stage systematically assesses whether the multi-tier design successfully meets the objectives defined in Sub-section 2.3.2, specifically concerning modularity, scalability, and performance efficiency. The evaluation is primarily based on comparing the architectural principles of the proposed multi-tier design against the limitations of the previous monolithic architecture. Table 4 serves as the primary evaluation instrument, where indicators such as separation of concerns, support for horizontal scaling, and efficiency of data access via caching are theoretically validated to ensure the design can sustainably support the nonformal learning environment. Although full empirical testing was not conducted, critical insight into the impact was obtained through this rigorous conceptual evaluation. This comparison provides theoretical proof that the artifact design directly addresses the root causes of inefficiency (such as high latency) previously identified as disrupting learning effectiveness. It is acknowledged that this conceptual evaluation is a limitation of the current study ; therefore, future work will include empirical testing post-implementation to measure the system's actual learning effectiveness using models such as the Technology Acceptance Model (TAM) or the DeLone & McLean Information System Success Model.

2.3.6 Communication Of Result

The final stage involves communicating the research findings and the derived technological artifact to both academic and practical audiences. The communication highlights the central theme of separating system responsibilities into four distinct tiers:

Presentation, Logic, Cache, and Data, as the key to achieving a scalable and manageable Learning Management System. The primary scientific contribution lies in the explicit conceptualization and integration of the Cache Tier as an independent and strategic architectural layer, offering a novel approach to optimizing performance and scalability in educational platforms. The practical contribution is the delivery of the comprehensive design blueprint to Woman School Serat Kartini, ensuring the system is theoretically capable of supporting future program expansion.

3. RESULT AND DISCUSSION

The multi-tier architecture design for the Learning Management System (LMS) of Woman School Serat Kartini has been fully developed and conceptually validated through the Design Science Research (DSR) methodology. The following subsections present the concrete artifacts, performance evaluations, strategic component analyses, functional validations, and broader implications of the proposed solution. These results demonstrate how the reconstructed system overcomes the limitations of the monolithic architecture while remaining deployable on low-cost shared hosting infrastructure.

3.1 Multi-Tier Architecture Design

As illustrated in Figure 2, the multi-tier design successfully reconstructs the LMS into four distinct and loosely coupled layers: Presentation, Logic, Cache, and Data Tier. This architecture achieves a high degree of modularity, allowing each layer to be developed, maintained, and scaled independently. The Presentation Tier (Next.js/React.js) handles the user interface, the Logic Tier (Express.js) manages business rules and acts as the API gateway, and the Data Tier (MySQL) serves as the persistent storage. This explicit separation addresses the inflexibility and development bottlenecks of the previous monolithic system.

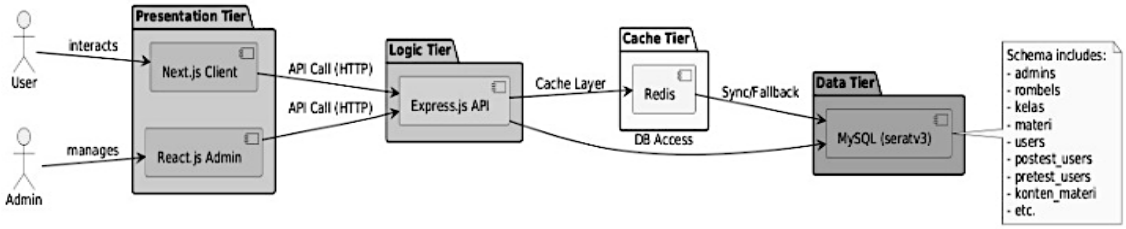

Figure 2. Multi-Tier Architecture Design

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

3.2 Database Modeling Design

The Data Tier utilizes MySQL as the primary storage engine, underpinned by a detailed relational database schema. This design is crucial as it represents the single source of truth for all system operations, interacting exclusively with the Logic Tier. As visualized in the Entity Relationship Diagram (ERD) in Figure 3, the comprehensive structure includes essential entities such as admins, rombel (classes), materi (materials), users, and evaluasi. These entities are linked through hierarchical and relational connections, for example, rombels to kelas and users to postest_users, effectively supporting the comprehensive academic structure and holistic learning tracking. The meticulous design of the Data Tier is fundamental to ensuring data integrity and optimizing the data retrieval processes required by the Logic Tier, especially since any query bottleneck here would necessitate the use of the Cache Tier to maintain system performance.

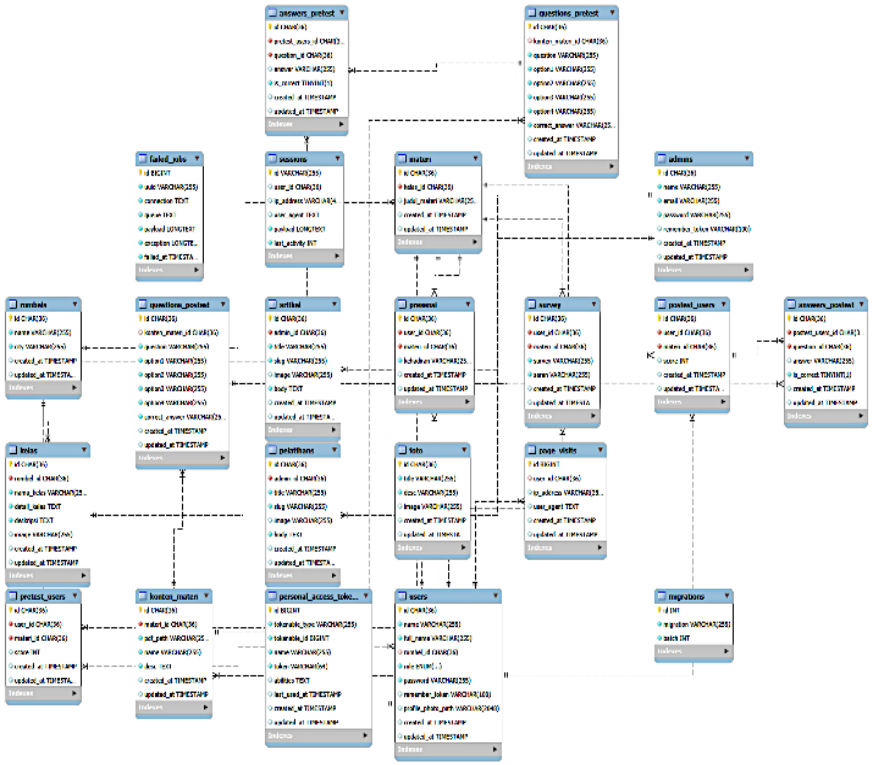

Figure 3. Entity Relationship Diagram

3.3 System Functionality Modeling

3.3.1 Actor dan Function Relationship

As illustrated in Figure 4, the Use Case Diagram visually validates the functional design, demonstrating clear interaction pathways between the primary actors (Visitor, Student,

and Admin) and the system's features. This model confirms that the multi-tier design is functionally accommodative, supporting core LMS scenarios like user registration, class access, material management, and administrative control through distinct and coherent interaction flows.

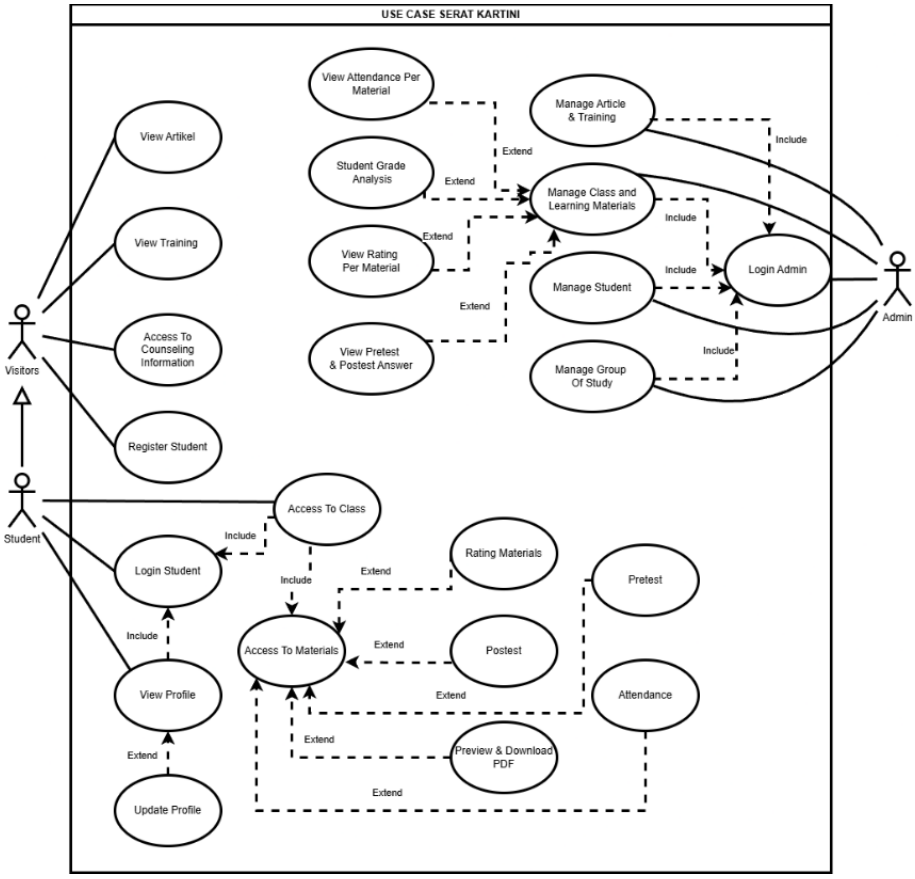

Figure 4. Use Case Diagram

3.3.2 Data Flow and Caching Strategy

The most critical aspect of the demonstration is the visualization of the data flow, particularly the implementation of the cache-aside strategy using Redis. The Sequence Diagram in Figure 5 provides a detailed step-by-step illustration of this process. This strategy is termed "cache-aside" as the cache (Redis) sits "aside" from the database, and the application's Logic Tier (API Server) assumes full responsibility for managing it—both by checking for data and manually populating the cache upon a miss. The flow initiates when the Client (Presentation Tier) requests data from the API Server (Logic Tier). Instead of directly querying the database, the API Server first attempts to retrieve the data from the Redis (Cache Tier). The alt (alternative) block within the diagram illustrates the two possible scenarios based on the cache's response:

1) Cache Hit: If the requested data (Get Key) is found within Redis ([Data in Cache]), the cache immediately returns the data (Return Cached Data) to the API Server, which then relays it to the Client. This path is exceptionally fast as it entirely bypasses any interaction with the main database (MySQL).
2) Cache Miss: If the data is not found in Redis ([Data Not Found]), the cache returns a Null response. The API Server must then proceed to query (Query Data) the primary MySQL (Data Tier) for the information. Once the data is retrieved (Return Data), the API Server performs two critical, simultaneous actions: it (a) stores a copy of this data into Redis (Store Data to Cache), ensuring that subsequent requests for the same data will result in a 'cache hit', and (b) returns the data to the Client to fulfill the initial request.

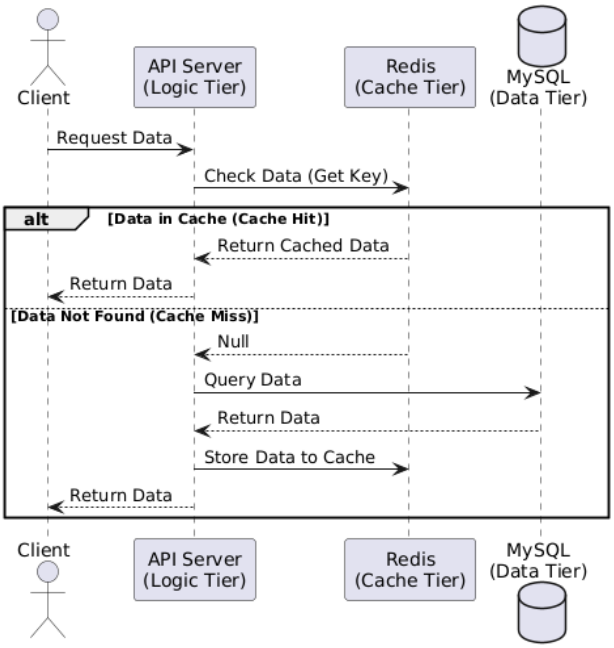

Figure 5. Cache Aside Strategy

This conceptual flow demonstrates how the strategy achieves a significant reduction in database queries and latency. By minimizing direct requests to the MySQL database, this model directly addresses the primary research objective: conceptually enabling the system to handle 500+ concurrent users and achieve the sub-100ms response time target. This is accomplished while still conserving the limited resources of the low-cost (under IDR 50,000/month) shared hosting environment, offering a technical solution to the critical >500ms latency bottlenecks inherent in the previous monolithic design.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

3.4 User Interface Visualization

To ensure the user interfaces shown in Figures 6 through 9 can load quickly and reliably, the backend caching strategy detailed in Figure 5 is employed. The visualization of the Presentation Tier, realized through high-fidelity mockups built with Next.js and React.js principles, demonstrates the user-centric output of the architectural design. These mockups cover key interfaces presented in the subsequent figures: Figure 6 depicts the responsive Homepage designed for public engagement; Figure 7 illustrates the Class Interface where students view available courses; Figure 8 shows the Detail Class Interface providing in-depth material access; and Figure 9 presents the comprehensive Dashboard Admin Interface for system management. Consequently, this visual presentation validates not only the system's technical responsiveness but also its capacity for inclusivity. By delivering a seamless and intuitive interface supported by the high-speed backend, the design effectively removes technological barriers for users with diverse digital literacy levels, thereby directly safeguarding learning continuity and supporting the empowerment goals of nonformal education in resource-constrained settings.

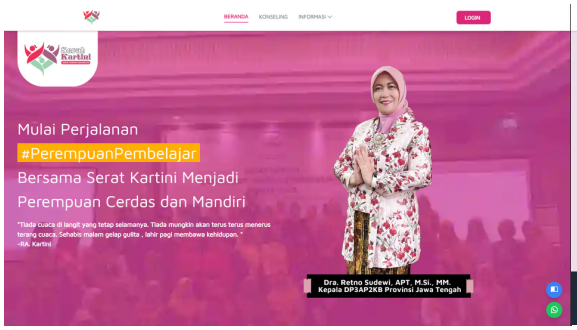

Figure 6. Homepage Interface

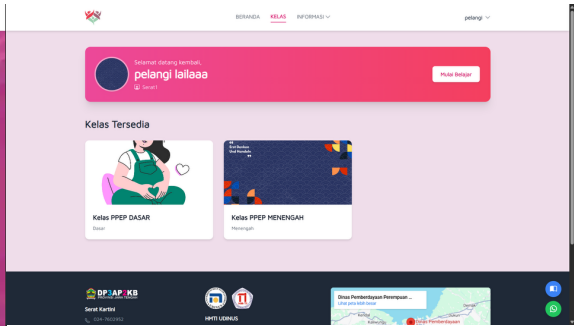

Figure 7. Class Interface

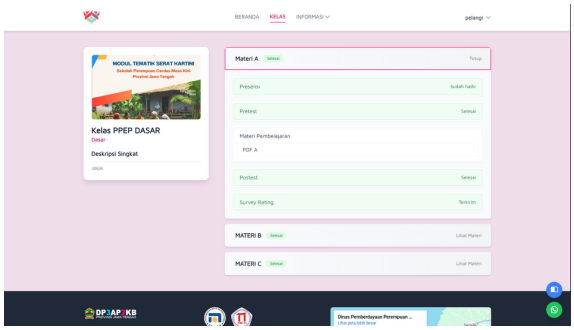

Figure 8. Detail Class Interface

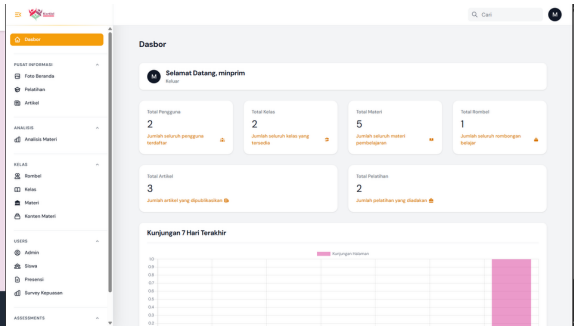

Figure 9. Dashboard Admin Interface

3.5 Discussion

The evaluation of the multi-tier architectural blueprint was performed as an ex-ante conceptual assessment, comparing the artifact's capabilities against the known limitations of the legacy monolithic system. The results of this comparison are summarized in Table 4. The analysis confirms that the proposed design fundamentally addresses the research objectives through enhanced modularity, superior scalability, and improved performance efficiency.

Table 4. Evaluation Result

No.	Evaluation Aspect	Assessment Description	Monolithic Architecture (Legacy)	Multi-Tier Architecture (Artifact)
1	Modularity	Separation of concerns across system components based on function and architectural tier.	Low (Tightly coupled components; single unit of code and deployment)	High (Separated into 4 independent tiers: Presentation, Logic, Cache, Data)
2	Scalability	System's ability to handle increases in users and features without performance degradation.	Limited; unable to fulfill horizontal scaling efficiently.	Supports independent scaling of each tier (horizontal scaling).
3	Data Access Efficiency	Efficiency in retrieving frequently accessed data and reducing database workload.	Fully dependent on MySQL; leads to high latency (exceeds 500 ms) during high traffic.	Redis Cache reduces query load, potentially lowering latency below 100 ms and supporting up to 500 concurrent users.

No.	Evaluation Aspect	Assessment Description	Monolithic Architecture (Legacy)	Multi-Tier Architecture (Artifact)
4	System Maintenance	Ease of development, debugging, and feature updates.	Complex (Changes affect the entire system; high risk of failure).	Easy (Components are independent, reducing risk of downtime).
5	Technology Flexibility	Ability to adopt modern frameworks and facilitate parallel development.	Rigid (Single technology stack).	High (Utilizes specialized modern stacks: Next.js, React, Express.js, Redis, MySQL).
6	Resource Efficiency (Shared Hosting)	System's ability to manage high load while minimizing consumption of shared infrastructure resources.	Poor; high resource consumption leads to overload and potential downtime.	High; Redis cache reduces database queries by an estimated 60-80%, conserving shared hosting resources.
7	Load Distribution Readiness	Support for distributing workload, including for widespread geographical deployment.	Limited (single server focus).	Supports horizontal scaling and load balancing across tiers, essential for regional expansion.

The reconstruction achieves a high degree of modularity by separating the system into four independent tiers. This design principle facilitates a clear separation of concerns, allowing each component to be developed, tested, and deployed in isolation. This eliminates the dependency bottlenecks of the monolithic architecture, drastically

simplifying maintenance and reducing the risk of system-wide failures during updates. Furthermore, the layered approach provides High Technology Flexibility (as shown in Table 4, No. 5) to integrate specialized modern technology stacks, such as Next.js, React, and Redis

The most significant validation lies in the system's enhanced Scalability and Data Access Efficiency. The multi-tier design inherently supports horizontal scaling (Table 4, No. 7) of components like the Presentation and Logic Tiers, allowing the system to handle increasing user loads. Performance is strategically optimized by the inclusion of the dedicated Cache Tier using Redis. As demonstrated in the Sequence Diagram (Figure 5), the cache-aside strategy directs high-volume, repetitive requests to Redis, which drastically reduces the number of queries to the MySQL primary database. This mechanism is crucial for mitigating the latency issues (which previously exceeded 500 ms) and supporting the goal of accommodating up to 500 concurrent users. This latency reduction is the primary technical 'insight' that theoretically translates to direct user impact: by eliminating the session interruptions and user frustration identified in the introduction, the design conceptually restores program participation and enhances learning effectiveness. This approach also proves highly efficient in Resource Efficiency (Table 4, No. 6), conserving the limited resources often found in shared hosting environments used by nonformal organizations.

Socially, this enhanced stability and adaptability carry significant implications for Woman School Serat Kartini's mission. By guaranteeing reliable performance and high Accessibility (Table 4, No. 3), the system removes a major technological barrier. This directly addresses the core problem of low learning effectiveness by mitigating the technical friction such as session interruptions and high latency that was previously identified as causing high module drop-off rates. This ensures that the LMS is more inclusive and sustainable for women participants with varied digital literacy and unstable internet access in rural areas, thereby supporting the program's expansion and empowerment goals.

This research offers substantial contributions across both theoretical and practical domains. Practically, the study delivers a validated Multi-Tier Architecture Blueprint for the reconstruction of the LMS. This blueprint is specifically designed for resource-

constrained environments, providing a highly scalable and responsive solution tailored to the operational demands of nonformal educational institutions. The selection of an efficient and lightweight technology stack (including Next.js, Express.js, and Redis) is crucial, as it ensures the system's stability and performance, enabling its successful and cost-effective deployment on shared hosting platforms. Theoretically, the key contribution lies in the explicit conceptualization and rigorous justification of the Cache Tier (Redis) as an independent, strategic, and integral layer within the multi-tier application architecture. This emphasis on a separate Cache Tier provides a novel approach to optimizing system performance and concurrency, offering a valuable reference point for the future design of technology artifacts in nonformal education contexts.

4. CONCLUSION

The redesign of the Learning Management System (LMS) for Woman School Serat Kartini successfully answers the research objectives by formulating a comprehensive multi-tier architecture blueprint that resolves the performance and scalability constraints of the prior monolithic system. The research discoveries confirm that dividing the system into four independent layers: Presentation, Logic, Cache, and Data that fulfills the requirement for modularity and isolated responsibilities. Specifically, the architectural design proves that the integration of a dedicated Cache Tier using Redis and a cache-aside strategy is the decisive solution for reducing system latency and alleviating query loads on the primary MySQL database.

The conclusions regarding the system's viability are derived from the ex-ante comparative evaluation, which validates the design against the research expectations for efficiency and inclusivity. Efficiency is theoretically evidenced by the capability of the Cache Tier to reduce relational database queries by an estimated 60-80%, thereby directly addressing the objective of optimizing limited shared hosting resources. Meanwhile, inclusivity is established through the projected ability of the system to maintain sub-100ms latency and high availability even under user surges. This technical stability serves as the prerequisite for accessibility, ensuring that women learners with unstable rural internet connections can participate without the barrier of session interruptions, thus achieving the program's empowerment goals.

For further development, this research offers several advices and suggestions. First, future work should proceed from this conceptual design to full software implementation and deployment to validate the architectural behavior in a live production environment. Second, subsequent research is advised to conduct empirical testing post-implementation to measure the system's actual impact on learning effectiveness. This should involve quantitative assessments using established frameworks such as the Technology Acceptance Model (TAM) or the DeLone & McLean Information System Success Model to rigorously evaluate user satisfaction and system adoption rates among the nonformal education participants.

REFERENCES

[1] R. Thinley and Gyeltshen, "A Theoretical Review on Non-Formal Education and Women Empowerment in Bhutan," *International journal of social science and human research*, vol. 7, no. 10, 2024, doi: 10.47191/ijsshr/v7-i10-91.

[2] F. Bugis, M. Kusuma Wirasti, and Y. Nurani, "Utilization of a Learning Management System to Develop Critical Thinking Skills," *Scaffolding: Jurnal Pendidikan Islam dan Multikulturalisme*, vol. 5, no. 2, pp. 243–255, Jun. 2023, doi: 10.37680/scaffolding.v5i2.2191.

[3] N. V. Thakre, "Learning System Application," *Gurukul International Multidisciplinary Research Journal*, vol. 12, no. 8, Jun. 2024, doi: 10.69758/gimrj2406i8v12p120.

[4] Amey Arun Padvekar and Vikaskumar Badriprasad Gupta, "Comparative Analysis of Monolithic vs. Distributed Architecture.," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 433–442, Jun. 2024, doi: 10.48175/ijarsct-18946.

[5] M. A. Z. Sidiq, M. I. Anshori, and R. A. Yaqin, "Penerapan Arsitektur Monolitik Pada Aplikasi Jasa Service Online Tekku Berbasis Web," *JUKI : Jurnal Komputer dan Informatika*, vol. 6, no. 1, pp. 27–36, May 2024, doi: 10.53842/juki.v6i1.418.

[6] H. J. Kim, K. P. Kim, and I. Jeong, "A joint development of an e-learning management system with the IAEA: Transitioning from an on-premises

hosting system to a cloud-based system," *Edelweiss Applied Science and Technology*, vol. 8, no. 6, pp. 6449–6458, 2024, doi: 10.55214/25768484.v8i6.3399.

[7] K. A. Saputri, B. Baharudin, A. F. Asyha, S. Bahri, I. F. Hasanah, and Q. Shabira, "Penggunaan Learning Management System (LMS) di Sekolah Menengah Pertama: A Systematic Literature Review," *LEARNING : Jurnal Inovasi Penelitian Pendidikan dan Pembelajaran*, vol. 4, no. 4, pp. 1264–1273, Dec. 2024, doi: 10.51878/learning.v4i4.4014.

[8] R. Irman Hermadi Yani Nurhadryani, "Analisis Uji Performa Aplikasi Dari Hasil Implementasi Refactoring Arsitektur Monolitik Ke Mikroservis dengan Decomposition dan Strangler Pattern," *Jurnal Sistem Cerdas*, vol. 6, no. 3, pp. 189–203, 2023, doi: 10.37396/jsc.v6i3.352.

[9] V. L. Nogueira, F. S. Felizardo, A. M. M. M. Amaral, W. K. G. Assuncao, and T. E. Colanzi, "Insights on Microservice Architecture Through the Eyes of Industry Practitioners," *ArXiv*, Aug. 2024, doi: 10.48550/arxiv.2408.10434.

[10] A. Schukin, N. Scerbakov, and E. Rezedinova, "Microservice Architecture of Modern eLearning Application," in *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, 2024, pp. 1–4. doi: 10.1109/CIEES62939.2024.10811386.

[11] M. V. Privalov and M. V. Stupina, "Improving web-oriented information systems efficiency using Redis caching mechanisms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 3, pp. 1667–1675, Mar. 2024, doi: 10.11591/ijeecs.v33.i3.pp1667-1675.

[12] M. I. Zulfa, A. Fadli, and A. W. Wardhana, "Strategi caching aplikasi berbasis in-memory menggunakan Redis server untuk mempercepat akses data relasional," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 2, pp. 157–163, Apr. 2020, doi: 10.14710/jtsiskom.8.2.2020.157-163.

[13] H. Schuldt, "Multi-Tier Architecture," in *Encyclopedia of Database Systems*, M. T. LIU LING and ÖZSU, Ed., Boston, MA: Springer US, 2009, pp. 1862–1865. doi: 10.1007/978-0-387-39940-9_652.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[14] M. A. Novianto and S. Munir, "Analisis dan Implementasi Restful API guna Pengembangan Sistem Informasi Akademik pada Perguruan Tinggi," *Jurnal Informatika Terpadu*, vol. 8, no. 1, pp. 47–61, Mar. 2022, doi: 10.54914/jit.v8i1.409.

[15] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.

[16] A. B. Raharjo, P. K. Andyartha, W. H. Wijaya, Y. Purwananto, D. Purwitasari, and N. Juniarta, "Reliability Evaluation of Microservices and Monolithic Architectures," in *2022 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, IEEE, Nov. 2022, pp. 1–7. doi: 10.1109/CENIM56801.2022.10037281.

[17] O. Pastor, A. Segooa, and J. I. Panach, "Teaching Design Science as a Method for Effective Research Development," 2024. doi: 10.48550/arxiv.2407.09844.