**Journal of Applied Informatics and Computing (JAIC)**
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861 

# Development of a Web-Based Learning Management System (LMS) to Support Employee Learning and Training

**Dini Aulia Fitri [1]*, Noper Ardi [2]****
* Informatics Engineering, Batam State Polytechnic
** Software Engineering Technology, Batam State Polytechnic
diniauliafitrii@gmail.com [1], noperardi@polibatam.ac.id [2]

**Article Info**

***Article history:***

Received …
Revised …
Accepted …

***Keyword:***

*Corporate training, Learning Management System, Next.js, PostgreSQL, Supabase, Usability, Waterfall.*

**ABSTRACT**

The rapid advancement of information technology has transformed corporate learning and training processes into strategic organizational priorities. Conventional learning platforms such as Moodle often present challenges including complex maintenance, high operational costs, rigid architecture, and limited customization for enterprise-specific workflows. This study aims to design and develop a web-based Learning Management System (LMS) using modern technologies—Next.js for the frontend and Supabase as a Backend-as-a-Service (BaaS) with PostgreSQL—to support standardized employee training in a multinational company (Company XYZ). The waterfall software development life cycle was applied, encompassing requirement analysis, system design, implementation, and testing. The developed system provides training management, multimedia content delivery, quiz-based evaluation, automated digital certification, analytics dashboards, and role-based access control (Employee, Admin Training, and Super Admin). Data security was implemented through Row Level Security (RLS) aligned with personal data protection principles. Usability evaluation was conducted using the System Usability Scale (SUS) with 20 respondents, yielding an average score of 79.125, categorized as Good (Acceptable) with Grade A according to Bangor et al. (2008). Functional testing confirmed that all core features operate according to specifications. The results demonstrate that the proposed system offers an integrated, usable, and organizationally adaptable solution for internal corporate training.



## I. INTRODUCTION

Digital transformation has significantly reshaped how organizations manage employee competency development. In multinational corporations, continuous and standardized training is essential to maintain workforce competence and align employee skills with evolving industry standards [1]. The learning process, which represents interaction between educators and learners, has shifted from conventional models to digital-based approaches. Organizations must adopt strategies aligned with information technology advancement to remain competitive in the global market [11].

Learning Management Systems (LMS) have become a primary technological solution for delivering, managing, and monitoring corporate learning activities in a centralized digital environment [2]. LMS functions as an e-learning platform enabling virtualization of teaching and learning through electronic technology. The system manages learning administration, organizes and stores materials, documents learning activities, and presents learning outcome reports digitally [2]. In corporate settings, LMS supports competency development programs that are measurable, repeatable, and scalable across multiple departments and geographic locations.

e-ISSN: 2548-6861

Despite widespread LMS adoption, many organizations still rely on conventional systems such as Moodle. Previous studies and industry experiences indicate that such platforms frequently suffer from monolithic architecture, outdated user interfaces, complex configuration, high maintenance costs, and limited flexibility for customization [3], [4]. Rahayu [3] found that Moodle-based e-learning applications require improved system stability, with some users still needing technical assistance. Syuhada and Handrianto [4] developed a web-based LMS using PHP and CodeIgniter but did not conduct usability evaluation or address security aspects comprehensively.

Zidan Nasih and Ali Mansur [1] demonstrated that LMS adoption positively influences digital transformation and employee digital competence development in industrial settings. However, their study focused on organizational impact rather than technical implementation details. Widjaja et al. [10] applied Next.js in a project management system with waterfall methodology and passed black-box testing, yet their system lacked external platform integration and flexibility for changing requirements. These research gaps motivated the present study to combine modern web technology stack implementation with formal usability evaluation in a corporate LMS context.

Company XYZ, a multinational organization operating in multiple countries, previously implemented a conventional LMS within its Information Technology (IT) department. The system proved insufficient in terms of maintainability, framework consistency, and adaptability to growing training requirements. The IT development division required a unified technology stack aligned with internal standards, specifically utilizing JavaScript-based frameworks that the development team already maintained for other internal applications.

Based on identified problems, this research addresses two primary questions: (1) How to redevelop a learning system that overcomes development, maintenance, and customization constraints caused by inconsistent frameworks in the previous platform? (2) How to evaluate the usability of the newly developed LMS to ensure it meets user needs and expectations? The objectives are to develop a web-based LMS using Next.js, Supabase, and PostgreSQL following the waterfall method, and to evaluate system usability using the System Usability Scale (SUS).

The scope is limited to registered employees of Company XYZ, without integration with external third-party platforms beyond core functional requirements. Technologies involved are restricted to Next.js, Supabase, and PostgreSQL. The remainder of this paper is organized as follows: Section II describes methodology and system design; Section III presents results and discussion; Section IV concludes the study.

Information technology in digital learning represents a fundamental shift in how knowledge transfer occurs within organizations. Nur [11] emphasizes that generational changes in learning systems require strategies aligned with technological advancement. In corporate environments, this translates to platforms that support self-paced learning, centralized content management, automated assessment, and verifiable certification at scale across multinational operations.

The selection of Next.js as frontend framework aligns with Company XYZ internal technology standards where JavaScript-based stacks dominate internal application development. Unified framework adoption across LMS and other internal tools reduces developer context switching, enables UI component reuse, and simplifies long-term maintenance compared to maintaining a separate PHP-based Moodle instance alongside modern JavaScript applications.

### *A. Related Research*

Summarizes previous research related to LMS development and usability evaluation, highlighting the research gap addressed by this study.

TABLE I
COMPARISON WITH PREVIOUS RESEARCH

| Researcher | Method / Technology | Advantages | Gap / Limitation |
|---|---|---|---|
| Rahayu [3] | SUS, Moodle, TAM | SUS effective; e-learning easy to use | System stability needs improvement |
| Zidan Nasih [1] | LMS, Path Analysis | Improves digital competence | No technical LMS integration discussion |
| Syuhada [4] | Waterfall, PHP, CodeIgniter | Easier material distribution | No usability or security evaluation |
| Widjaja [10] | Next.js, Waterfall, Appwrite | Real-time Kanban; passed Black Box | Not integrated; rigid waterfall |
| This study | Next.js, Supabase, SUS | Modern stack; SUS score 79.125 | Limited to 20 respondents; no load test |

## II. METHOD

### *A. Research Approach*

This study applied the waterfall software development model, a linear and sequential approach introduced by Winston Royce in 1970, in which each phase must be completed before proceeding to the next [5]. The waterfall method is characterized by non-iterative process flow where each phase must be finalized before continuing. Although considered traditional, waterfall remains widely applied in software engineering practice, particularly for projects with clearly defined requirements and minimal expected changes during implementation [5].

The development process consisted of five phases: (1) requirement analysis, (2) system design, (3) implementation, (4) verification (testing), and (5) maintenance. Requirement gathering was conducted through interviews and observations

with the IT development team and training administrators at Company XYZ. The IT department's internal system development division served as the primary stakeholder, providing insights into existing workflow pain points and desired feature specifications.

During the requirement phase, functional needs such as login, training material management, evaluation, certification, and employee progress reporting were identified alongside non-functional requirements including usability, security, availability, and maintainability. Design phase produced use case diagrams, activity diagrams, ERD, and wireframes. Implementation utilized Next.js, Supabase, and PostgreSQL. Verification included black-box functional testing and SUS usability evaluation with 20 respondents.

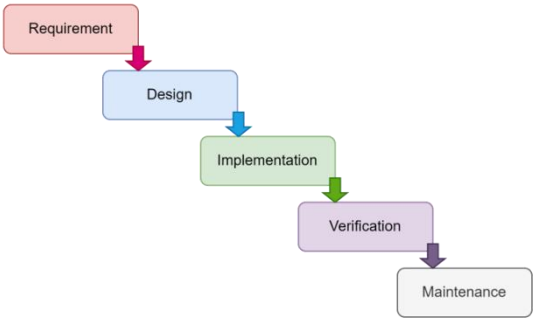


Figure 1. Waterfall development methodology applied in this study

Table II summarizes activities conducted in each waterfall phase during LMS development at Company XYZ.

TABLE II
WATERFALL DEVELOPMENT PHASES

| Phase | Activities Conducted | Deliverables |
|---|---|---|
| Requirement | Interviews with IT team and trainers; observation of existing workflow | FR-01 to FR-19, non-functional requirements |
| Design | Use case, activity diagrams, ERD, wireframes in Figma/Draw.io | System design documents, database schema |
| Implementation | Next.js frontend, Supabase backend, module development | Functional LMS modules |
| Verification | Black-box testing, SUS evaluation with 20 respondents | Test reports, SUS score 79.125 |
| Maintenance | Bug fixes, input validation, security improvements | Updated stable system version |

Development tools included a laptop with AMD Ryzen 3 5300U processor, 8 GB RAM, and 512 GB storage running Windows 11. Software tools comprised Visual Studio Code as IDE, Figma and Draw.io for design, Postman for API testing, and GitHub for version control. Research materials included Company XYZ training process documents, digital training modules, dummy user data for simulation, SUS questionnaire results, and academic literature on digital learning and usability evaluation.

### *B. System Architecture and Technology Stack*

The LMS was built using a modern three-tier architecture separating presentation, application logic, and data layers. This separation enables independent scaling, maintenance, and technology updates for each layer without affecting the entire system.

The frontend layer utilizes Next.js, a React-based framework supporting server-side rendering (SSR), static site generation (SSG), and optimized performance for responsive web interfaces [6]. Next.js enables faster page loads through SSR, improved SEO capabilities, and simplified deployment workflows. Tailwind CSS was used for responsive styling ensuring optimal display across desktop, tablet, and mobile devices. The App Router architecture organizes pages into route groups for public, employee, and admin interfaces.

The backend layer employs Supabase, an open-source Backend-as-a-Service (BaaS) providing integrated authentication, real-time PostgreSQL database, file storage, and edge functions [7]. Supabase serves as an open-source alternative to Firebase with PostgreSQL as its relational database foundation. Row Level Security (RLS) enables granular data access control at the table row level, implementing least privilege principles. This eliminates the need for building and maintaining custom backend servers, reducing infrastructure complexity and accelerating development timelines.

The data layer uses PostgreSQL, an open-source relational database management system designed for complex, large-scale workloads [8]. PostgreSQL provides advanced features including transactions, subqueries, triggers, views, and materialized views. Performance experiments demonstrate efficient processing on datasets up to one million records, making it suitable for enterprise information systems requiring high reliability and speed.

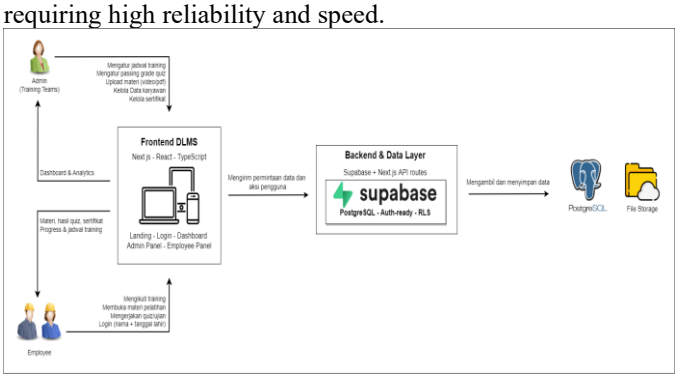


Figure 2. General system architecture of the developed LMS

### *C. Functional and Non-Functional Requirements*

Nineteen functional requirements (FR-01 to FR-19) were defined through stakeholder interviews. Table III presents the complete functional requirements specification.

TABLE III
FUNCTIONAL REQUIREMENTS

| Code | Requirement Description | Actor |
|---|---|---|
| FR-01 | Login to system | Admin, Employee |
| FR-02 | Logout from system | Admin, Employee |
| FR-03 | View dashboard | Admin, Employee |
| FR-04 | Manage profile | Admin, Employee |
| FR-05 | View notifications | Admin, Employee |
| FR-06 | View training list | Employee |
| FR-07 | Enroll in training | Employee |
| FR-08 | Access training materials | Employee |
| FR-09 | Track training progress | Employee |

e-ISSN: 2548-6861

| | | |
|---|---|---|
| FR-10 | Complete quiz evaluation | Employee |
| FR-11 | View training schedule | Employee |
| FR-12 | View and download certificate | Employee |
| FR-13 | Manage training data | Admin |
| FR-14 | Manage training materials | Admin |
| FR-15 | Manage employee data | Admin |
| FR-16 | Manage training schedules | Admin |
| FR-17 | Manage quizzes and questions | Admin |
| FR-18 | Manage certificates | Admin |
| FR-19 | View analytics and reports | Admin |

Non-functional requirements ensure system quality attributes beyond functional capabilities. Table IV defines availability, reliability, ergonomics, portability, response time, safety, security, maintainability, and language requirements.

TABLE IV
NON-FUNCTIONAL REQUIREMENTS

| Parameter | Requirement |
|---|---|
| Availability | Accessible at all times except scheduled maintenance |
| Reliability | Consistent operation without fatal errors during training |
| Ergonomics | User-friendly interface for diverse user backgrounds |
| Portability | Usable across desktop, tablet, and mobile devices |
| Response Time | Stable access with multiple concurrent users |
| Safety | Protection from data loss during operations |
| Security | Data confidentiality via RLS; UU PDP compliance |
| Maintainability | Modular architecture for easy updates |
| Language | English interface throughout the system |

### *D. System Design*

System design artifacts included use case diagrams, nineteen activity diagrams, Entity-Relationship Diagrams (ERD), and wireframes created using Figma and Draw.io. The use case diagram identifies three actors (Employee, Admin Training, Super Admin) interacting with nineteen primary use cases covering the complete training lifecycle from enrollment through certification.

The database schema encompasses entities: profiles (user accounts with role assignments), trainings (course metadata with category, status, thumbnail), materials (multimedia content with type and ordering), enrollments (employee-training relationships with progress tracking), material_progress (individual material completion records), schedules (training session timing and location), quizzes (evaluation settings with timer and passing grade), questions (multiple choice question bank), quiz_attempts (employee quiz results), certificates (digital certification records), and notifications (system alerts).

Authentication design uses full name and date of birth matched against the profiles table. Security layers include input validation, rate limiting, server-side verification, hidden password derivation, JWT session management, route middleware protection, and RLS database policies. Admin routes are protected at both application middleware and database policy levels to prevent unauthorized access.

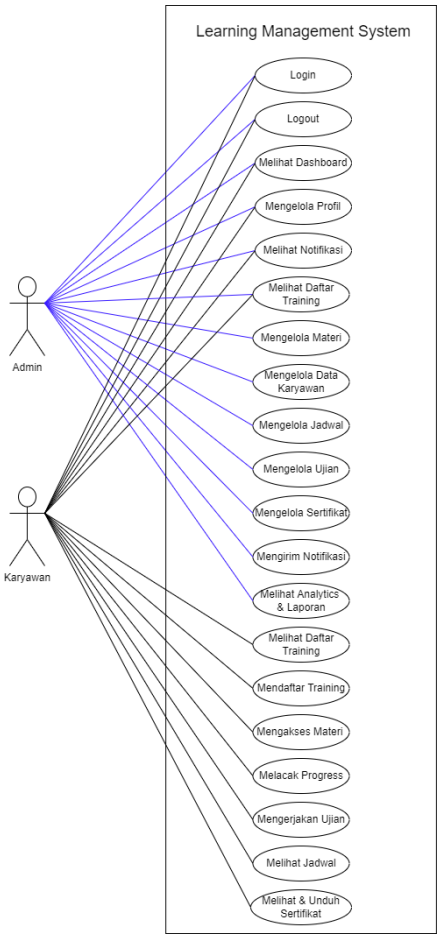


Figure 3. Use case Diagram of the LMS

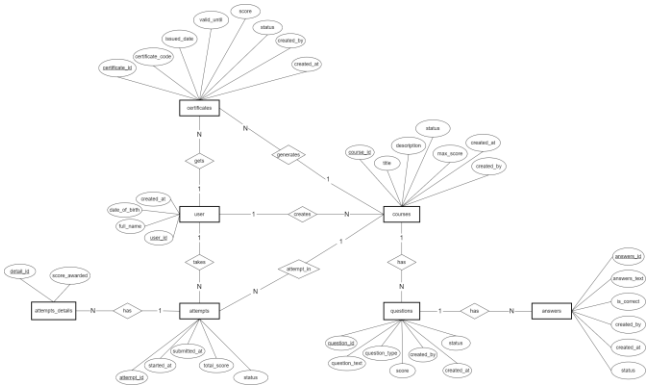

Figure 4. Entitiy-Relationship Diagram of the LMS

Table V describes primary database entities and their relationships within the PostgreSQL schema managed through Supabase.

TABLE V
DATABASE ENTITY DESCRIPTION

| Entity | Primary Attributes | Relationship |
|---|---|---|
| profiles | id, name, email, role, date_of_birth | One-to-many with enrollments |
| trainings | id, title, category, status, thumbnail | One-to-many with materials, quizzes |
| materials | id, title, type, content_url, order | Many-to-one with trainings |

| | | |
|---|---|---|
| enrollments | id, user_id, training_id, progress, status | Links profiles and trainings |
| material_progress | id, enrollment_id, material_id, completed | Tracks individual material completion |
| schedules | id, training_id, start_time, end_time, location | Many-to-one with trainings |
| quizzes | id, training_id, timer, passing_grade | One-to-many with questions |
| questions | id, quiz_id, question_text, correct_answer | Many-to-one with quizzes |
| quiz_attempts | id, user_id, quiz_id, score, status | Stores evaluation results |
| certificates | id, enrollment_id, issue_date, certificate_no | Generated upon completion |
| notifications | id, user_id, message, is_read | System alerts for users |

Nineteen activity diagrams were created documenting system workflows including login, material upload, schedule creation, passing grade configuration, certificate upload, progress monitoring, evaluation result viewing, material access, material download, schedule viewing, quiz taking, evaluation result checking, certificate viewing, certificate download, and logout. Wireframes were designed for login, home, dashboard, training material, schedule management, evaluation test, certificate management, and training monitoring interfaces.

The login use case (FR-01) exemplifies system interaction design. Users access the login page and enter full name and date of birth. The system validates credentials against the profiles table via Supabase authentication service. Upon successful validation, a JWT session token is stored in browser storage and the user is redirected to the role-appropriate dashboard. Alternative flows handle invalid credentials with error notification, empty field validation, connection failures with Supabase service, and incomplete environment configuration alerts. This design prioritizes simplicity for corporate users while maintaining security through server-side verification rather than client-only validation.

Row Level Security policies were configured for each database table to enforce access control at the data layer. Example policies include employees can SELECT their own enrollment records where user_id matches authenticated profile; admins can INSERT/UPDATE/DELETE training records where profile role equals admin_training or super_admin; quiz_attempts are readable only by the attempt owner and assigned administrators. This multi-layer security approach combining application middleware, JWT authentication, and database RLS provides defense in depth aligned with enterprise data protection requirements and Indonesia's Personal Data Protection Law (UU PDP) principles.

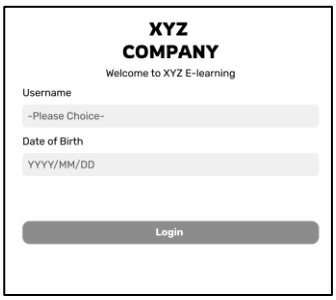

Figure 5. System wireframe Login

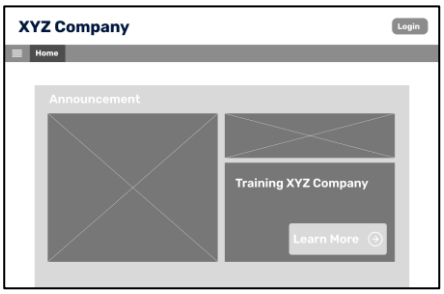

Figure 6. System wireframe home page

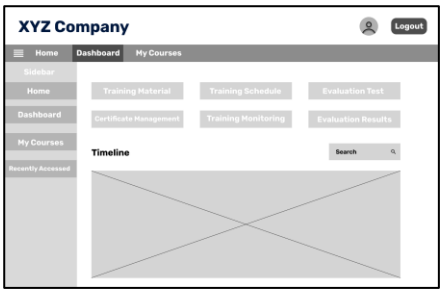

Figure 7. System wireframe dashboard

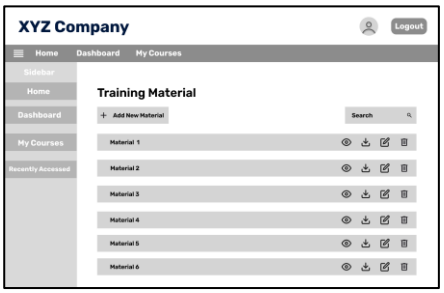

Figure 8. System wireframe training material

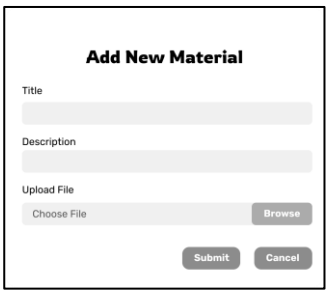

Figure 9. System wireframe add material

e-ISSN: 2548-6861

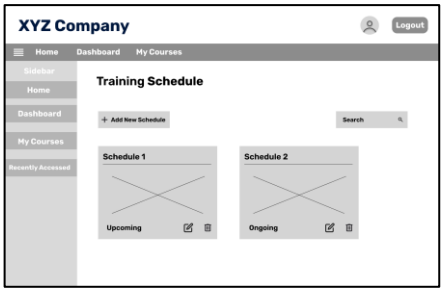


Figure 10. System wireframe training schedule

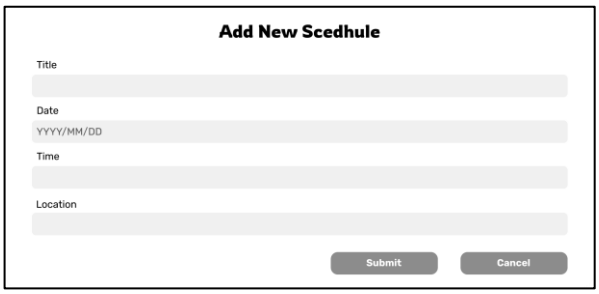


Figure 11. System wireframe add schedule

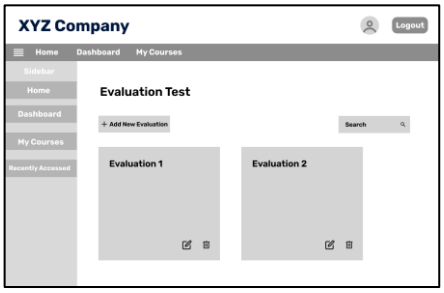


Figure 12. System wireframe evaluation test

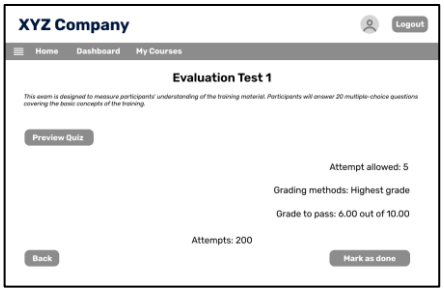


Figure 13. System wireframe quiz evaluation (admin)

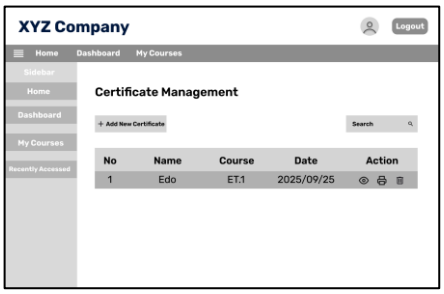


Figure 14. System wireframe certificate management

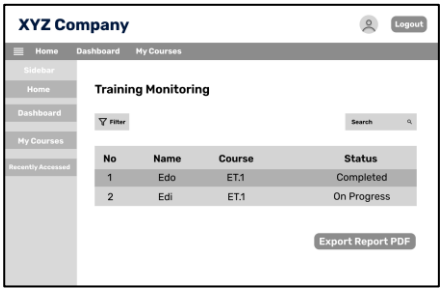


Figure 15. System wireframe training monitoring

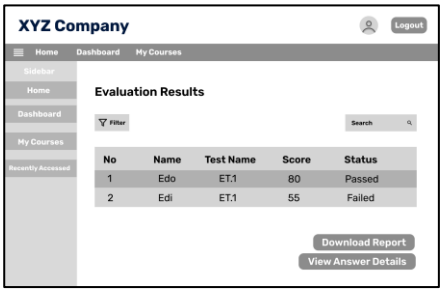


Figure 16. System wireframe evaluation results

### *E. Implementation Details*

#### *1) Public and Authentication Interface*

The landing page serves as the initial access point, built with Next.js App Router and Tailwind CSS. It displays platform identity through hero section, digital learning illustration, and benefits overview. The login page implements simplified authentication using full name and date of birth, with server-side validation, rate limiting, JWT sessions, and route protection middleware ensuring secure access before dashboard redirection based on user role.

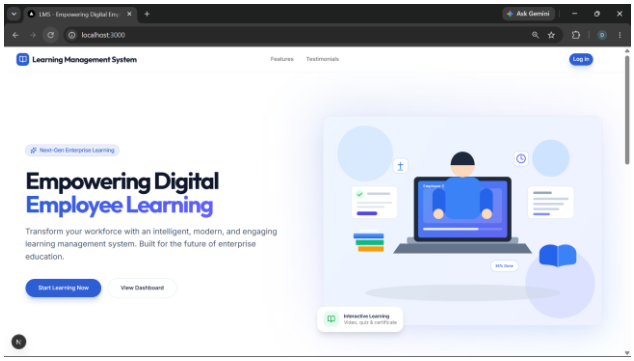


Figure 17. Landing page

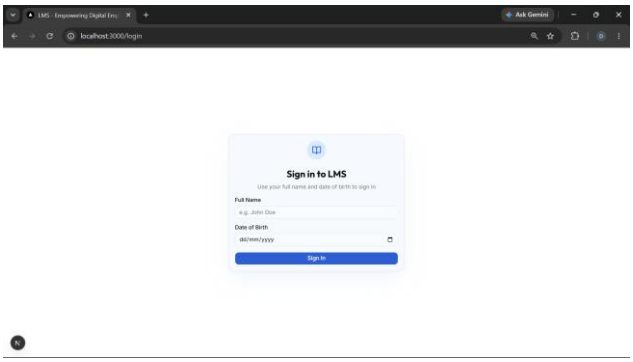


Figure 18. Login interface

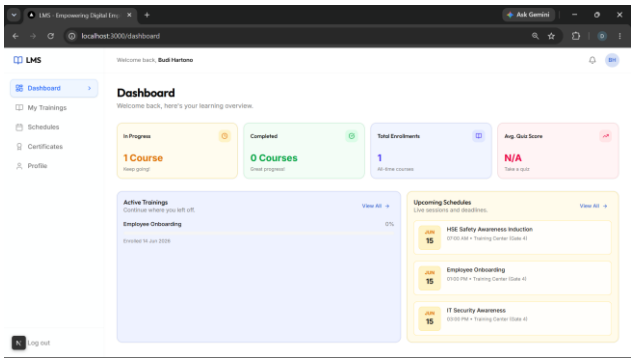


Figure 19. Employee dashboard

### *2) Employee Interface Modules*

The employee dashboard aggregates data from enrollments, quiz_attempts, and schedules through parallel Supabase queries, displaying active trainings, completed trainings, total enrollments, average quiz scores, active training cards, and upcoming schedule summaries in a single unified view.

My Trainings page presents available trainings as cards containing title, category, thumbnail, and individual progress percentage. Category tab navigation and search bar enable efficient training discovery. Training detail pages provide curriculum sidebar showing material sequence with completion status indicators.

Material delivery supports multimedia formats: video content through YouTube Embed integration and PDF documents through Google PDF Viewer embedded directly in the learning page. Progress completion uses Mark as Complete functionality, updating material_progress table in real-time and recalculating overall enrollment progress percentage.

Quiz evaluation displays multiple choice questions progressively with client-side countdown timer implemented via React state. Answers persist in component state during the session. Upon submission or timeout, the system calculates scores, compares against administrator-defined passing grades, stores results in quiz_attempts, and displays pass/fail status immediately.

Schedule page presents training calendars and upcoming event lists from schedules table with training title, category, time, location, and deadline information. Certificate page lists earned certifications with PDF preview and download using jsPDF client-side generation containing participant name, training title, issue date, and unique certificate number.

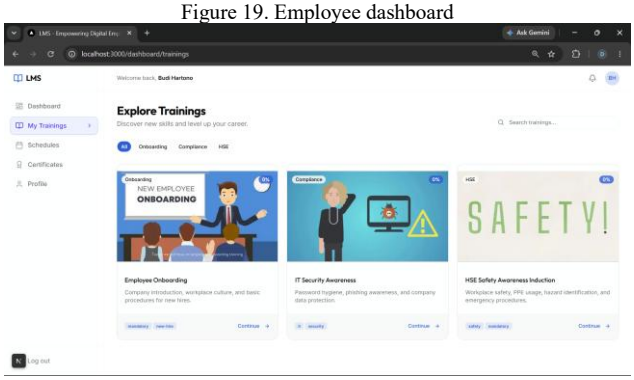


Figure 20. My Trainings page

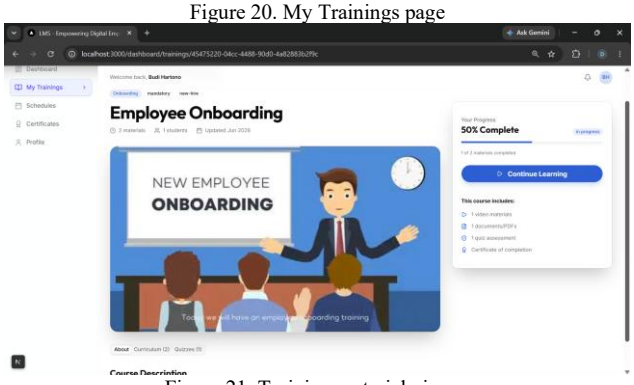


Figure 21. Training material viewer

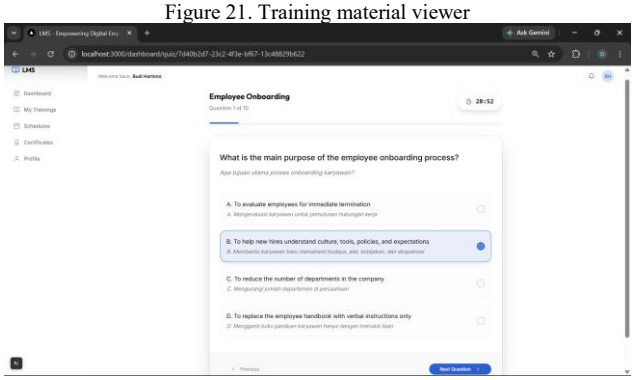


Figure 22. Quiz evaluation page

e-ISSN: 2548-6861

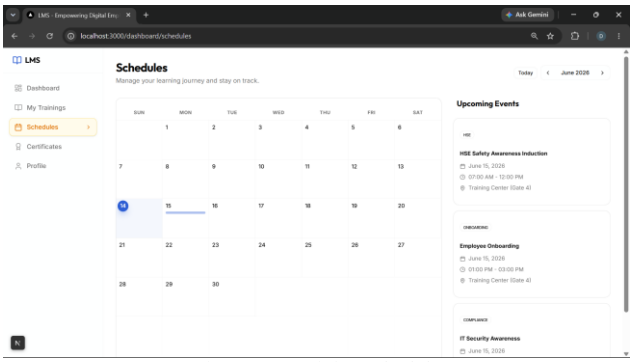


Figure 23. Training schedule

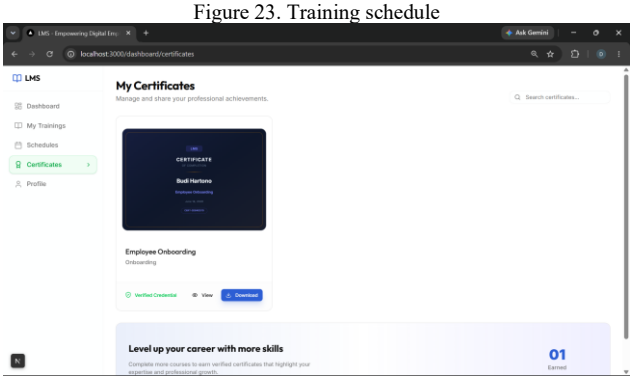


Figure 24. Certificate pages

### *3) Administrator Interface Modules*

Manage Trainings provides CRUD operations through table view with form dialogs. Zod schema validation ensures data integrity for title, description, category, tags, publication status, and thumbnail URL before Supabase Client writes to trainings table. RLS policies restrict write operations to admin-role users.

Manage Materials groups content by training with tab navigation filter. Each material record contains title, type (video/PDF/document), content URL, and display order. External content links (YouTube, Google Drive) are stored as URLs rather than uploaded files, reducing storage requirements.

Manage Users displays registered users with summary statistics (total users, admins, employees). Admin Training can view, edit, and create users. Super Admin additionally deletes accounts except their own. Manage Quizzes links evaluations to specific trainings with configurable timer, passing grade, and question bank management supporting multiple choice format with designated correct answers.

Analytics dashboard aggregates user counts, enrollment statistics, completion rates, average quiz scores, top trainings by enrollment, and individual learner performance metrics through parallel Supabase queries with client-side aggregation for real-time reporting.

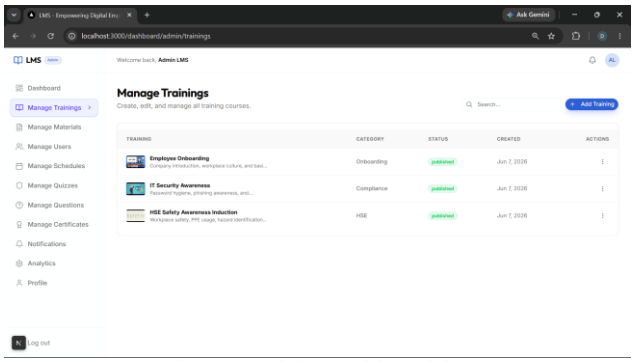


Figure 25. Admin panel for training

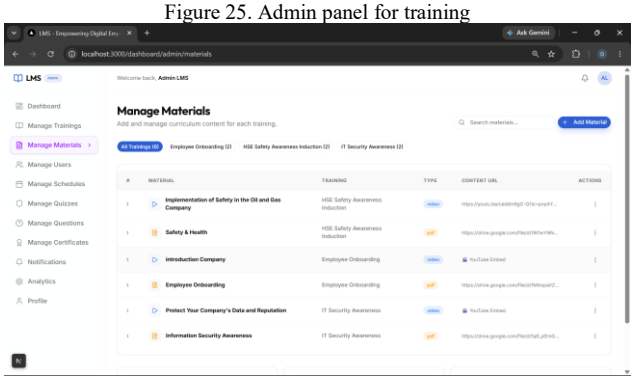


Figure 26. Material management

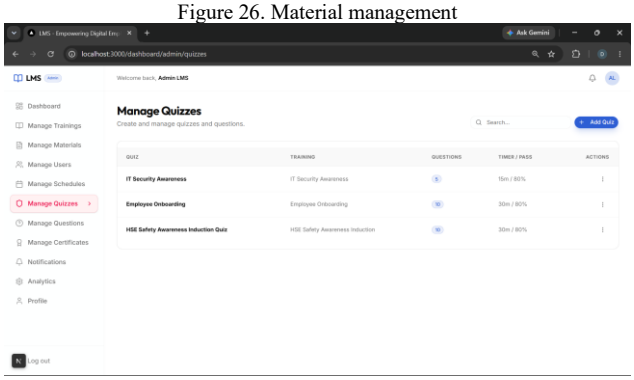


Figure 27. Admin quiz management

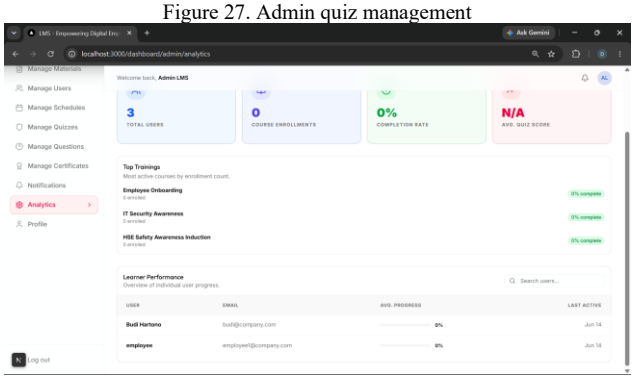


Figure 28. Analytics dashboard

## *F. Testing and Evaluation Methods*

### *1) Functional Testing*

Functional testing employed black-box testing methodology, verifying system behavior against defined functional requirements without examining internal code

structure. Test scenarios covered authentication, training CRUD operations, material upload and access, quiz completion with score calculation, certificate generation and download, and role modification by Super Admin. Each test case documented expected results, actual results, and pass/fail status.

***2) Usability Evaluation (SUS)***

Usability evaluation employed the System Usability Scale (SUS), a standardized 10-item questionnaire using 5-point Likert scale developed by Brooke [9]. SUS provides a reliable and valid usability measurement producing a composite score from 0 to 100, interpretable through acceptability ranges and grade scales established by Bangor et al. [12].

Twenty respondents from Company XYZ participated: administrators managing training data and employees using the LMS for learning activities. Each respondent completed assigned tasks (login, dashboard navigation, training enrollment, material completion, certificate download) before filling the SUS questionnaire. Table VI lists the ten SUS statements used.

Scoring procedure: odd items (1,3,5,7,9) contribution = response minus 1; even items (2,4,6,8,10) contribution = 5 minus response. Total contribution multiplied by 2.5 yields individual SUS score. Average across all respondents provides overall usability metric.

TABLE VI
SUS QUESTIONNAIRE STATEMENTS

| No. | SUS Statement |
|---|---|
| 1 | I think I would like to use this system frequently. |
| 2 | I found the system unnecessarily complex. |
| 3 | I thought the system was easy to use. |
| 4 | I think I would need technical support to use this system. |
| 5 | I found the various functions in this system were well integrated. |
| 6 | I thought there was too much inconsistency in this system. |
| 7 | I would imagine that most people would learn to use this system very quickly. |
| 8 | I found the system very cumbersome to use. |
| 9 | I felt very confident using the system. |
| 10 | I needed to learn a lot of things before I could get going with this system. |

Table VII presents an example SUS score calculation for a single respondent, demonstrating the scoring procedure applied to all twenty participants.

TABLE VII
EXAMPLE SUS SCORE CALCULATION

| No. | Statement | Response (1-5) | Contribution |
|---|---|---|---|
| 1 | Would like to use frequently (positive) | 4 | 4 - 1 = 3 |
| 2 | Unnecessarily complex (negative) | 2 | 5 - 2 = 3 |
| 3 | Easy to use (positive) | 5 | 5 - 1 = 4 |
| 4 | Need technical support (negative) | 3 | 5 - 3 = 2 |
| 5 | Functions well integrated (positive) | 4 | 4 - 1 = 3 |
| 6 | Too much inconsistency (negative) | 2 | 5 - 2 = 3 |
| 7 | Learn quickly (positive) | 5 | 5 - 1 = 4 |
| 8 | Very cumbersome (negative) | 3 | 5 - 3 = 2 |
| 9 | Very confident (positive) | 4 | 4 - 1 = 3 |
| 10 | Need to learn a lot (negative) | 2 | 5 - 2 = 3 |
| Total contribution | | | 30 |
| SUS Score (30 × 2.5) | | | 75 |

For the complete evaluation, total raw contribution from all 20 respondents was 633 points. Average SUS = (633 × 2.5) / 20 = 79.125. Respondents included both Admin Training personnel who manage system content and employees who consume training materials, ensuring evaluation covered both administrative and end-user perspectives.

Prior to SUS evaluation, each respondent completed a structured task sequence: (1) login using full name and date of birth; (2) navigate dashboard and identify active trainings; (3) browse available training catalog and enroll in one training; (4) access and complete at least one training material; (5) attempt quiz evaluation if available; (6) view and download certificate if eligible. This task-based approach ensures SUS responses reflect actual usage experience rather than first impressions alone.

Table VIII documents the respondent composition for SUS evaluation, ensuring transparency in participant selection and role distribution across administrative and end-user categories.

TABLE VIII
SUS RESPONDENT COMPOSITION

| No. | Respondent Role | Primary Tasks Evaluated | SUS Score Range |
|---|---|---|---|
| 1-5 | Admin Training | Manage trainings, materials, users, analytics | 72 - 85 |
| 6-15 | Employee | Enroll, access materials, complete quiz | 75 - 88 |
| 16-18 | Employee (IT Dept) | Full workflow including certificates | 78 - 82 |
| 19-20 | Admin Training (Senior) | All admin functions + role management | 80 - 87 |
| Average (n=20) | | Combined all tasks | 79.125 |

### *G. Security Architecture*

Security architecture implements defense-in-depth strategy across three layers: application middleware, authentication service, and database Row Level Security. Application middleware intercepts all protected route requests, verifying JWT token validity and extracting user role before rendering role-specific pages. Invalid or expired tokens trigger automatic redirect to login page preventing unauthorized dashboard access.

Authentication service validates user credentials through server-side API routes rather than direct client-database communication. Rate limiting prevents brute force login attempts by restricting authentication request frequency per IP address. Password derivation from date of birth occurs server-side using cryptographic hashing, ensuring raw credentials never persist in client-side storage or network transmission logs.

Database RLS policies enforce row-level access control independent of application logic. Even if application middleware is bypassed through direct API calls, RLS ensures users access only authorized data rows. Admin Training role receives write permissions on content management tables while Employee role receives read-only access to published training content and write access only to personal enrollment and quiz attempt records.

Compliance with Indonesia Personal Data Protection Law (UU PDP) is addressed through data minimization (collecting only name, email, date of birth, and role), purpose limitation (data used exclusively for training management), and access control (RLS ensuring users access only their authorized data). Audit trail through database timestamps on all records

e-ISSN: 2548-6861

supports accountability requirements for corporate training documentation.

## III. RESULT AND DISCUSSION

### *A. Implementation Results*

The LMS was successfully deployed with all planned modules operational in a production environment accessible to Company XYZ employees. All nineteen functional requirements were implemented and verified through systematic testing procedures described in the following sections.

The landing page implementation utilizes Next.js App Router with server-side rendering for optimal initial page load performance. Hero section communicates platform value proposition while responsive Tailwind CSS styling ensures consistent appearance across device form factors. Login authentication validates user credentials against profiles table with multi-layer security including rate limiting to prevent brute force attempts and JWT session tokens with configurable expiration.

The employee dashboard successfully aggregates enrollment, quiz, and schedule data through optimized parallel Supabase queries, reducing page load time compared to sequential data fetching. Dashboard widgets display: count of active trainings currently in progress, total completed trainings, overall enrollment count, and average quiz score across all attempted evaluations. Active training cards provide quick navigation to in-progress courses while upcoming schedule section highlights imminent training sessions with date, time, and location details.

My Trainings interface organizes available courses using card-based layout with visual thumbnail, category badge, title, and circular progress indicator. Tab navigation filters trainings by category (e.g., Technical, Soft Skills, Compliance) while search functionality enables text-based discovery across titles and descriptions. Enrolled trainings display progress percentage calculated as ratio of completed materials to total materials in curriculum.

Material delivery module supports heterogeneous content types through unified viewer interface. Video materials embed YouTube player directly within learning page eliminating context switching to external platforms. PDF materials render through Google PDF Viewer enabling in-browser reading without file downloads. Curriculum sidebar displays sequential material list with checkmark indicators for completed items and current position highlighting for navigation orientation.

Real-time progress tracking updates material_progress records immediately upon Mark as Complete action, providing accurate progress percentages across dashboard and training detail views. When all materials reach completion status, enrollment status transitions to completed, triggering certificate eligibility evaluation based on quiz performance if applicable.

Quiz module reliably implements countdown timers with automatic submission upon timeout. Question presentation uses progressive disclosure showing one question at a time to reduce cognitive load. Answer selection persists in React component state throughout session duration. Upon submission, scoring algorithm calculates percentage correct, compares against passing grade threshold, assigns pass/fail status, and persists complete attempt record including individual question responses in quiz_attempts table.

Certificate generation produces PDF files with accurate participant information and unique identification numbers derived from database record IDs. jsPDF library enables client-side PDF creation without server-side document processing overhead. Generated certificates include formal layout with participant full name, training title, completion date, and verifiable certificate number suitable for internal HR documentation purposes.

Admin analytics provides actionable insights through aggregated metrics computed from parallel database queries. Key performance indicators include total registered users, active enrollments, overall completion rate percentage, and average quiz score across organization. Top Trainings ranking identifies most popular courses by enrollment volume while Learner Performance table displays individual employee progress metrics enabling targeted intervention for struggling learners.

Super Admin role management enables dynamic role assignment between Employee, Admin Training, and Super Admin levels. Role changes persist to profiles table and take effect through RLS policy re-evaluation upon user re-authentication. UI menu structure adapts to assigned role ensuring users access only authorized functional areas, reducing accidental exposure of administrative capabilities to regular employees.

Table IX compares key characteristics between the previous Moodle-based LMS and the newly developed system, highlighting improvements achieved through redevelopment.

TABLE IX
COMPARISON PREVIOUS VS DEVELOPED LMS

| Aspect | Previous (Moodle) | Developed LMS |
|---|---|---|
| Frontend Framework | PHP-based Moodle theme | Next.js with React |
| Backend Architecture | Monolithic self-hosted | Supabase BaaS |
| Database | MySQL/MariaDB | PostgreSQL with RLS |
| UI Customization | Limited theme options | Fully custom Tailwind UI |
| Maintenance | Complex plugin management | Unified JS stack |
| Security | Plugin-dependent | RLS + JWT + middleware |
| Usability Score | Not evaluated | SUS 79.125 (Grade A) |

### *B. Functional Testing Results*

Black-box testing confirmed all seven primary test scenarios passed successfully, validating core system functionality against specification requirements. Table X presents detailed testing results

TABLE X
BLACK-BOX TESTING RESULTS

| No | Test Scenario | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| 1 | User login with valid credentials | Redirect to role-based dashboard | As expected | Valid |
| 2 | Admin creates new training | Data saved and listed | As expected | Valid |
| 3 | Admin uploads material (Video/PDF) | Content accessible to enrolled users | As expected | Valid |
| 4 | Employee completes quiz | Score calculated and displayed | As expected | Valid |
| 5 | System issues certificate | PDF generated when score ≥ passing grade | As expected | Valid |
| 6 | Employee downloads certificate | Correct PDF with participant details | As expected | Valid |
| 7 | Super Admin changes user role | Access updated on next session | As expected | Valid |

Additional functional tests were conducted for edge cases including invalid login credentials rejection, duplicate enrollment prevention, quiz timeout automatic submission, certificate generation blocked when quiz score below passing grade, and unauthorized admin page access redirection for employee role users. All edge case tests produced expected behavior confirming robust error handling throughout the application.

### *C. Usability Evaluation Results*

SUS evaluation with 20 respondents produced comprehensive usability metrics summarized in Table III. Total raw contribution score across all respondents was 633 points. Applying the SUS formula (total × 2.5 / n), the average score is 79.125.

According to Bangor et al. [12] interpretation framework, score 79.125 falls within three evaluation dimensions: (1) Acceptability Range — Acceptable, indicating the system can be accepted by users; (2) Adjective Rating — Good, approaching Excellent; (3) Grade Scale — Grade A (range 75–90), representing very good usability quality.

These results indicate the LMS is perceived as usable, intuitive, and acceptable by end users with varying digital literacy levels. The score validates design decisions regarding interface consistency, logical navigation flow, and integrated feature presentation tailored to Company XYZ organizational context.

TABLE XI
SUS EVALUATION SUMMARY

| Parameter | Value |
|---|---|
| Number of Respondents | 20 |
| Total Raw Contribution Score | 633 |
| Total SUS Score (633 × 2.5) | 1,582.5 |
| Average SUS Score | 79.125 |
| Acceptability Range | Acceptable |
| Adjective Rating | Good |
| Grade Scale | A (75–90) |

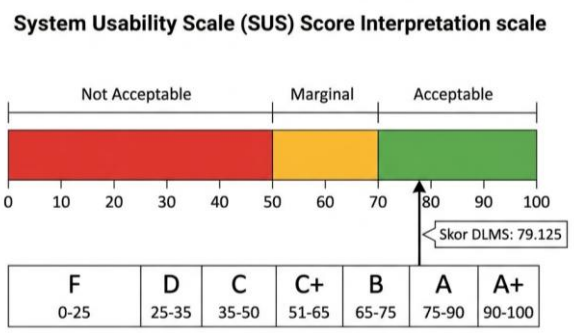

Figure 29. SUS score interpretation based on Bangor et al. (2008)

### *D. Discussion*

The redevelopment of Company XYZ LMS using Next.js and Supabase successfully addressed limitations of the previous Moodle-based platform. Modern architecture eliminated framework inconsistency across internal applications, reduced backend infrastructure complexity through BaaS adoption, and enabled responsive interface design customized for organizational workflows rather than generic open-source templates.

Compared with Rahayu [3], who obtained positive SUS results for Moodle-based e-learning but noted stability concerns, this study demonstrates that modern architecture choices can achieve comparable usability scores (79.125) while providing better maintainability through unified technology stack. Syuhada and Handrianto [4] developed LMS functionality without usability metrics; this research provides empirical SUS evidence supporting design quality claims.

Zidan Nasih and Ali Mansur [1] established LMS organizational value for digital competence; this study complements their findings with technical implementation evidence and usability validation. The combination of Next.js frontend performance optimization, Supabase backend simplification, and PostgreSQL data reliability creates a cohesive platform suitable for enterprise internal training requirements.

Row Level Security implementation represents a significant security design decision aligned with Indonesia's Personal Data Protection Law (UU PDP) principles. RLS ensures role-based data isolation at the database layer, complementing application-level access controls. Employees can only access their own enrollment and quiz data, while administrators access management functions appropriate to their assigned roles.

Study limitations include: usability evaluation limited to 20 respondents within a single IT department; formal penetration testing and load testing not conducted; operational cost analysis not performed; external platform integration (HR systems, SSO providers) not implemented. These limitations define scope boundaries and suggest directions for future research and development iterations.

Despite limitations, findings provide practical contribution for organizations seeking to modernize internal LMS platforms. The documented architecture, requirement specifications, testing procedures, and usability metrics offer replicable framework for similar corporate LMS development initiatives using contemporary web technologies.

From an organizational perspective, the developed LMS enables Company XYZ to standardize internal training processes previously distributed across email, shared drives, and ad-hoc meeting sessions. Centralized training catalog, automated progress tracking, and digital certification reduce administrative overhead while improving audit trail completeness for compliance reporting.

Technical contribution of this research lies in demonstrating feasibility of BaaS architecture (Supabase) for enterprise LMS deployment without dedicated backend development team. Frontend developers familiar with React/Next.js can implement complete LMS functionality leveraging Supabase auto-generated APIs, authentication services, and RLS policies.

The SUS score of 79.125 positions the developed system above the industry average SUS score of approximately 68 reported by Bangor et al. [12], indicating above-average usability performance. Positive scores on items related to ease of use, function integration, and learning speed suggest the interface design successfully minimized complexity while maintaining feature completeness.

## IV. CONCLUSSION

This study successfully developed a web-based Learning Management System for corporate employee training at Company XYZ using Next.js, Supabase, and PostgreSQL with waterfall development methodology. All nineteen functional requirements were implemented and validated through black-box testing with seven test scenarios achieving Valid status.

Usability evaluation using SUS with 20 respondents produced average score 79.125 (Good/Acceptable, Grade A per Bangor et al. [12]), confirming the system is user-friendly and suitable for supporting internal training activities. The LMS provides integrated platform for training management, multimedia delivery, quiz evaluation, progress monitoring, and digital certification.

Row Level Security incorporation ensures data protection consistent with personal data privacy principles. Future work recommendations include: (1) Supabase Realtime notifications for training updates and quiz deadlines; (2) React Native or PWA mobile access; (3) expanded multi-department usability evaluation; (4) formal security auditing and load testing; (5) SCORM content format support; (6) long-term infrastructure cost-benefit analysis.

The practical implications of this research extend beyond Company XYZ to any organization seeking cost-effective LMS modernization. BaaS architecture reduces initial development investment by eliminating custom backend API development while PostgreSQL ensures data model flexibility for evolving training requirements. Organizations with existing React/Next.js development capabilities can adapt the documented architecture pattern with minimal additional infrastructure provisioning.

Academically, this study contributes empirical evidence combining modern web technology implementation with standardized usability evaluation in corporate LMS context. Future researchers can extend this work through longitudinal usability studies, comparative analysis with other BaaS platforms, integration of learning analytics algorithms for personalized training recommendations, and investigation of gamification elements to enhance employee engagement in mandatory compliance training programs.

## ACKNOWLEDGMENT

The author expresses gratitude to the Informatics Engineering Department and Software Engineering Technology Study Program of Politeknik Negeri Batam, supervising lecturer Mr. Noper Ardi, S.Pd., M.Eng., IT Manager Mr. Wahyu Hidayat and Assistant IT Manager Mr. Ikhsan Kurniawan of Company XYZ, and all respondents who participated in the SUS usability evaluation.

## REFERENCES

[1] M. Zidan Nasih and S. Ali Mansur, "Digital Transformation: The Effect of Learning Management Systems in Developing Employee Digital Competence (Study at A Chemical Company in Gresik)," J. Ekon. Bisnis, vol. 18, no. 2, 2024.

[2] I. Widya, P. Pratomo, and R. Wahanisa, "Pemanfaatan Teknologi Learning Management System (LMS) di Unnes Masa Pandemi Covid-19," Seminar Nas. Hasil Penelit. Unnes, vol. 7, no. 2, pp. 547-554, 2021, doi: 10.15294/snhunnes.v7i2.730.

[3] N. Rahayu, "Analisis UX Pada Aplikasi E-Learning Menggunakan Metode SUS (System Usability Scale)," J. Komputer, vol. 3, no. 2, 2025.

[4] F. A. Syuhada and Y. Handrianto, "Perancangan Aplikasi Learning Management System Berbasis Web Pada Trustco Cipta Madani," J. Komputer Antartika, vol. 1, 2023.

[5] A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," ResearchGate, 2020.

[6] J. P. Joarno et al., "Implementasi Progressive Web Apps Pada Website Gethelp Menggunakan Next.js," J. Teknol. Inf. Kharisma, 2022.

[7] Supabase Inc., "Supabase Documentation: Row Level Security," 2024. [Online]. Available: https://supabase.com/docs/guides/auth/row-level-security

[8] A. Dwi Praba and M. Safitri, "Studi Perbandingan Performansi Antara MySQL dan PostgreSQL," J. Teknol. Inf., vol. VIII, no. 2, 2020.

[9] J. Brooke, "SUS: A quick and dirty usability scale," in Usability Evaluation in Industry, P. W. Jordan et al., Eds. London: Taylor & Francis, 1996, pp. 189-194.

[10] F. A. Widjaja, B. Hakim, and C. Author, "Sistem Manajemen Proyek Berbasis Website Dengan Metode Kanban," CO-SCIENCE J. Comput. Sci., vol. 5, no. 2, 2025.

[11] T. Nur, "Strategi Pembelajaran Era Digital," in Proc. Annu. Conf. Islam. Educ. Soc. Sci., vol. 1, no. 2, 2019.

[12] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," Int. J. Hum-Comput. Interact., vol. 24, no. 6, pp. 574-594, 2008,

[13] Vercel Inc., "Next.js Documentation: Server-Side Rendering," 2024. [Online]. Available: https://nextjs.org/docs