Renode LMS Platform Technical Documentation

Document owner: Renode Technologies (Pty) Ltd
Document status: Publication-ready business and technical documentation
Effective date: 24 May 2026
Applicable platform: Renode LMS Platform / OpenCampus LMS
Primary jurisdiction: Republic of South Africa

Executive Summary

The Renode LMS Platform is a cloud-hosted, white-label learning management system designed for primary schools, secondary schools, tertiary institutions, TVET colleges, skills development providers, government programmes and enterprise academies. The platform enables each institution to operate its own branded LMS environment while sharing a common secure application, database and infrastructure foundation.

The platform supports course delivery, learner enrolment, assessments, online classes, messaging, discussion forums, calendar management, reporting, sponsor visibility, multilingual user interfaces, institution branding, support ticketing and provider administration. It is designed as a multi-tenant SaaS system, where each institution sees only its own users, courses, content, support records and configuration.

The intended audience for this document includes investors, enterprise clients, government stakeholders, developers, compliance reviewers, security assessors and internal operational staff.

Platform Overview

Purpose

The purpose of the Renode LMS Platform is to help institutions deliver, manage and monitor digital learning at scale without building separate learning infrastructure for each school, college, university or training provider. The system is built for white-label deployment, meaning each institution can present its own logo, colour scheme, domain, language preferences and selected layout style to its users.

Core Functionality

Capability	Description
Institution onboarding	Subscription-based registration, payment activation and licence tracking.
User management	Admin-managed creation, import, editing, password reset and deactivation of users.
Course management	Course folders, modules, chapters, sections, subsections, PDF resources, videos and SCORM package storage.
Assessments	Quizzes, tests, exams and assignments with objective auto-marking and manual grading.
Online classes	Jitsi Meet, Microsoft Teams, Google Meet or external meeting link routing.
Messaging	Direct communication between permitted users.
Forums	Institution-scoped discussion categories, posts and comments.
Calendar	Private, learner, teacher and shared agenda management.
Reporting	Grade visualisation, XLSX export and CSV/XLSX import.
Sponsor portal	Visibility of sponsored learners’ grades and progress.
LMS provider portal	Support ticket management, customer assistance history, language editing and layout template management.

Key Benefits

- Reduced time to launch a branded LMS for an institution.
- Multi-role access model for administrators, teachers, learners, sponsors and provider staff.
- Institution-level data separation and branding.
- South African language support with translation management.
- Secure asset handling for institution files and course material.
- Support for enterprise integrations including PayFast, Jitsi, Microsoft Teams, Google Meet, Google Cloud Storage and cloud-hosted MySQL.
- Reporting and performance visibility for education leadership and sponsors.

User Ecosystem

```mermaid
flowchart LR
Provider["LMS Provider Users"] --> Institution["Institution Tenant"]
Institution --> Admin["Admin Users"]
Institution --> Teacher["Teacher Users"]
Institution --> Learner["Learner Users"]
Institution --> Sponsor["Sponsor Users"]
Admin --> Course["Courses and Content"]
Teacher --> Course
Learner --> Course
Sponsor --> Reports["Learner Performance Reports"]
Provider --> Tickets["Support Tickets and Customer Assistance"]
```

High-Level Architecture

```mermaid
flowchart TB
Browser["User Browser"] --> Web["Linode/Akamai Dedicated Web Server"]
Web --> PHP["PHP Application Layer"]
PHP --> MySQL["Google Compute MySQL Instance"]
PHP --> GCS["Google Cloud Storage for Institution Media"]
PHP --> Redis["Google Cloud Redis Cache"]
Web --> LB["Akamai Load Balancer / CDN"]
PHP --> PayFast["PayFast Subscription Payments"]
PHP --> Meet["Jitsi / Microsoft Teams / Google Meet"]
PHP --> Mail["SMTP / Transactional Email"]
```

The current implementation is a PHP and MySQL application served through Apache/XAMPP locally and intended for Linux/Apache or Nginx/PHP-FPM in production. The platform uses server-rendered PHP pages, progressive AJAX enhancement, secure sessions, CSRF protection, prepared SQL statements and institution-scoped database queries.

User Role Documentation

Teacher Users

Role Description

Teacher users are institution staff members responsible for course delivery, learner

engagement, online classes, assessments and grading.

Permissions and Access Levels

Area	Access
Courses	Create, read, update and delete assigned or created courses, subject to institution policy.
Course content	Add modules, chapters, sections, subsections, PDFs, videos, YouTube links and SCORM packages.
Assessments	Create assessments, tests and exams; mark submissions; allow resubmissions.
Online classes	Create or join classes using Jitsi, Microsoft Teams, Google Meet or approved external links.
Reports	View grading visualisations for assigned courses and learners.
Messaging	Send and receive messages with admin users, teachers, learners and sponsors where enabled.
Forum	Manage categories, posts and comments within role limits.
Calendar	Create private or learner-visible agenda items.

Dashboard Overview

The teacher dashboard presents courses, course content, assessments, submissions, online classes, messages, forum discussions, calendar items, reports and profile settings.

Typical Workflows

1. Log in with institution-issued credentials or institution SSO where enabled.
2. Create or update a course.
3. Add course modules, chapters and learning material.
4. Schedule an online class and invite learners.
5. Create an assessment with a due date.
6. Review learner submissions and grade manual answers.
7. Export marks for reporting.

Notifications and Communication Tools

Teachers use messaging, class chat, forum comments and calendar visibility to communicate with learners and institutional staff.

Security Considerations

Teachers must only access learners and course records within their institution. The platform enforces session timeouts, role checks, CSRF validation and institution-scoped database access.

Reporting Capabilities

Teachers can view bar-chart performance summaries, submission records and assessment scores for courses within their teaching scope.

Sponsor Users

Role Description

Sponsor users are guardians, bursary sponsors, employer sponsors or authorised stakeholders who need visibility into learner performance.

Permissions and Access Levels

Area Access
--- ---
Sponsored learners Read-only visibility of assigned learner performance.
Profile Read and update own profile details, language preference and profile picture.
Reports View grades and performance summaries for sponsored learners.

Dashboard Overview

The sponsor dashboard focuses on learner progress, average scores, assessment participation and profile settings.

Typical Workflows

1. Log in to the sponsor portal.
2. Review assigned learners.
3. Check assessment participation and performance.
4. Update profile details and preferred language.

Security Considerations

Sponsor access must be explicitly mapped by an admin user. Sponsors must not see unrelated learners, courses, messages or private institutional content.

Learner Users

Role Description

Learner users consume course material, attend online classes, submit assessments and participate in permitted communication channels.

Permissions and Access Levels

Area Access
--- ---
Courses Read enrolled course content.
Course progress Mark course content complete and navigate previous/next content.
Assessments Submit answers before due dates or within approved resubmission windows.
Online classes Join classes they are invited to or enrolled in.
Messaging Communicate with permitted users.
Forum Read categories and manage own posts/comments subject to moderation.
Calendar Manage own agenda and view shared learner events.
Reports View own grades and submissions.

Dashboard Overview

The learner dashboard includes enrolled courses, structured course content, assessments, online classes, messages, forums, calendar and grade reports.

Typical Workflows

1. Log in or use SSO where configured.
2. Open an enrolled course.
3. Move through modules, chapters, sections and subsections.

4. View PDFs and videos through protected LMS views.
5. Join scheduled online classes.
6. Submit assessments before the due date.
7. Review automatic marks for objective questions.

Security Considerations

The platform restricts learner content to enrolled courses. PDF and training content is served through controlled asset endpoints. Browser-level screenshot prevention cannot be guaranteed by any web application, but the platform uses deterrent controls such as protected viewing surfaces, direct-download restrictions and controlled asset access.

Admin Users

Role Description

Admin users represent the institution customer. They manage users, branding, courses, content, enrolments, assessments, support tickets, calendars, forums, language preferences and institutional configuration.

Permissions and Access Levels

Area	Access
Users	Create, import, edit, deactivate, delete non-admin users and reset user passwords.
Branding	Update logo, colour scheme, domain, layout template and SSO configuration.
Courses	Manage courses, teachers, learners and course content.
Assessments	Create, update, delete and grade assessments.
Online classes	Create classes and route them through Jitsi, Teams, Google Meet or external meeting links.
Support	Submit support tickets to the LMS provider and track status.
Reports	Import/export marks and view performance visualisations.
Forums and calendar	Manage institution-wide categories, posts and agenda visibility.

Dashboard Overview

The admin dashboard is the primary tenant management interface. It exposes user management, branding, sponsors, support, courses, assessments, online classes, messages, forums, calendar, reports and profile tools.

Typical Workflows

1. Register institution and complete subscription payment.
2. Log in after PayFast payment activation.
3. Configure logo, colours, domain, layout template and optional SSO.
4. Import teachers, learners, sponsors and support roles from CSV/XLSX.
5. Create courses and assign teachers/learners.
6. Add assessments and class schedules.
7. Monitor performance and export reports.
8. Log support requests to the provider portal.

Security Considerations

Admin access is powerful and must be protected with strong passwords, SSO where available, session expiry, least-privilege operational procedures and regular access reviews.

LMS Provider Users

Role Description

LMS Provider Users are Renode Technologies staff or authorised contractors who support customers, manage platform-level configurations and assist with operational administration.

Provider Roles

Provider Role	Description
Super Admin	Full provider portal access, provider user creation, ticket handling and customer assistance.
Support Agent	Customer support, ticket updates and secure reset-link assistance.
Sales	Customer and licence visibility for commercial follow-up.
Finance	Subscription, licence and payment-related support.
UI/UX Designer	Layout template creation and dashboard design updates.
Language Editor	Translation and language file management.

Dashboard Overview

The provider dashboard includes live support ticket updates, customer admin details, assistance history, expiry reminder email actions, reset-link issuance, layout template uploads, translation management and provider user management.

Security Considerations

Provider users operate across tenants and must be subject to stricter internal controls, including role-based access, audit logging, MFA where deployed, secure reset-link handling and regular access reviews.

Technical Documentation

Frontend Architecture

The public website uses static HTML, CSS and JavaScript. Public pages load the shared navbar and footer from ‘index.html‘ through ‘site_shared.js‘, so updates to public navigation and footer links propagate to pages such as Legal, Docs and Book a Demo.

The LMS dashboards are PHP-rendered HTML pages enhanced by ‘lms_app.js‘. The dashboard uses AJAX form submission and section switching to reduce page refreshes. UI sections are controlled by URL query parameters and body classes.

Backend Architecture

The backend is implemented in PHP with MySQLi prepared statements. Core platform logic is centralised in ‘lms_bootstrap.php‘, authentication helpers are in ‘security.php‘, database connection is in ‘db_conn.php‘, and dashboard rendering is in ‘lms_dashboard.php‘.

Database Structure

Core tables include:

Table	Purpose
institutions	Tenant records, branding, licence, payment and SSO settings.

users Institution users and roles.
courses Institution-scoped courses and course folders.
course_enrollments Learner-to-course mapping.
course_teachers Teacher-to-course mapping.
course_content_items Modules, chapters, sections, PDFs, videos, lessons and SCORM records.
assessments Tests, exams, assignments and due dates.
assessment_questions Question types and correct answers.
assessment_submissions Learner submissions and scores.
messages Institution-scoped messaging.
online_classes Class sessions, meeting providers and joining URLs.
online_class_media Uploaded recordings and videos.
forum_categories, forum_posts, forum_comments Discussion forum data.
calendar_events Agenda and visibility records.
sponsor_learners Sponsor-to-learner mapping.
support_tickets Customer support requests.
provider_users Renode provider portal users.
demo_requests Public demo interest submissions.

Authentication and Authorisation Flow

1. User submits credentials or SSO provider callback returns a verified identity.
2. The system verifies the user against institution, status and licence rules.
3. On success, the session is regenerated and role-specific session values are set.
4. The user is routed to the correct dashboard.
5. Each dashboard request enforces role permissions and institution scoping.
6. Sessions expire after five minutes of inactivity by default.

API Structure

The application uses PHP endpoints rather than a standalone REST framework. Important endpoints include:

Endpoint Purpose
--- ---
login.php Institution user login.
sso_start.php SSO provider redirect initialisation.
sso_callback.php OAuth2/OIDC callback handling.
provider_api.php Provider portal AJAX actions.
secure_asset.php Authenticated institution asset delivery.
notify.php PayFast payment notification validation.
signup.php Institution signup and PayFast subscription initiation.

File Storage Handling

The platform stores institution assets under private storage configured by ‘.env‘.

Environment Variable Purpose
--- ---
APP_STORAGE_DRIVER ‘local‘ for local storage, ‘gcs‘ for Google Cloud Storage when implemented.
APP_LOCAL_STORAGE_ROOT Private local storage path.
APP_GCS_BUCKET Google Cloud Storage bucket name.
APP_GCS_PROJECT_ID Google Cloud project identifier.
APP_GCS_PUBLIC_BASE_URL Optional CDN/public base URL for signed delivery.
APP_GCS_CREDENTIALS_JSON Service account credential path or JSON secret reference.

Direct public access to institution assets is blocked. Authenticated access is served through ‘secure_asset.php‘ with tenant validation.

Security Implementation

The platform implements:

- Argon2id password hashing with SHA-512 pre-hashing.
- CSRF tokens on state-changing actions.
- Prepared SQL statements for database access.
- Role-based authorisation.
- Institution-scoped data filtering.
- Secure session cookie settings.
- Five-minute inactivity logout.
- Upload type and MIME validation.
- Protected asset delivery.
- PayFast signature and payment validation.
- Reset tokens stored as hashes with expiry.

Multi-Tenant Logic

Each institution is a tenant. Tenant isolation is enforced by ‘institution_id‘ fields and scoped queries. Courses, users, assessments, calendar events, forums, messages, support tickets and assets are tied to an institution.

User Session Handling

Sessions are started through ‘app_secure_session_start()‘. Login regenerates the session ID. Activity timestamps are updated on authenticated access. Inactive sessions expire after the configured timeout.

Error Handling

Errors are surfaced to users as dashboard flash messages or redirects. Security failures return appropriate HTTP responses or redirect to login. Production deployments should log errors centrally while suppressing sensitive details from browser output.

Logging and Monitoring

Recommended production logging includes:

- Authentication success and failure events.
- Provider reset-link actions.
- Support ticket status changes.
- Payment notifications and validation results.
- Upload validation failures.
- Application exceptions.
- Web server access and error logs.
- Database performance logs.

Scalability Considerations

The target production architecture supports:

- Akamai load balancing and CDN acceleration.
- Linode/Akamai dedicated web application servers.

- Google Compute MySQL database instance.
- Google Cloud Redis cache instance.
- Google Cloud Storage for multimedia and institution content.
- Horizontal scaling of PHP application nodes.
- Centralised logging and monitoring.

Developer Documentation

Folder Structure

Path Description
--- ---
/ Public PHP/HTML entry points and core app files.
/config Path configuration and environment-derived constants.
/docs/markdown Source documentation in Markdown.
/docs/pdf Generated PDF documents.
/layout_templates Provider-managed CSS layout templates.
/storage Private local storage for institution assets.
/institutions Legacy public institution storage, blocked by ‘.htaccess‘.

Environment Variables

Variable Description
--- ---
APP_ENV Application environment, for example ‘local‘, ‘staging‘ or ‘production‘.
APP_URL Public base URL.
APP_STORAGE_DRIVER Storage driver, currently ‘local‘; intended ‘gcs‘ for production storage integration.
APP_LOCAL_STORAGE_ROOT Local private storage root.
APP_SESSION_TIMEOUT_SECONDS Session inactivity timeout.
DB_HOST MySQL host, expected to point to Google Compute in production.
DB_PORT MySQL port.
DB_NAME Database name.
DB_USER Database username.
DB_PASS Database password.
PAYFAST_SANDBOX ‘true‘ or ‘false‘.
PAYFAST_MERCHANT_ID PayFast merchant ID.
PAYFAST_MERCHANT_KEY PayFast merchant key.
PAYFAST_PASSPHRASE PayFast passphrase.
SMTP_HOST SMTP host.
SMTP_PORT SMTP port.
SMTP_USER SMTP username.
SMTP_PASS SMTP password.
SMTP_FROM Default sender address.

Deployment Process

1. Provision Linode/Akamai dedicated web server.
2. Configure Apache or Nginx with PHP-FPM.
3. Deploy application code outside user-writable directories.
4. Configure ‘.env‘ with production database, PayFast, SMTP and storage values.
5. Configure HTTPS certificates.
6. Configure Akamai load balancer and CDN rules.
7. Provision Google Compute MySQL database.
8. Provision Google Cloud Redis cache where session or cache offloading is implemented.
9. Provision Google Cloud Storage bucket for multimedia and course assets.

10. Run database schema bootstrap or migrations.
11. Validate PayFast production callback endpoint.
12. Validate SSO providers and redirect URIs.
13. Run security tests and backup verification.

Local Development Setup

1. Install XAMPP or equivalent PHP/MySQL stack.
2. Place the codebase under ‘C:\xampp\htdocs\LMS‘.
3. Start Apache and MySQL.
4. Configure ‘.env‘ for local database settings.
5. Open ‘http://localhost/LMS/‘.
6. Import or initialise ‘white_label_lms‘.
7. Use ‘php -l‘ to lint changed PHP files.

Third-Party Integrations

Integration Purpose
--- ---
PayFast Subscription payment and auto-renewal initiation.
Jitsi Meet Embedded online class sessions.
Microsoft Teams External online class joining.
Google Meet External online class joining.
Google Cloud Storage Intended production storage for course media and institution assets.
Google Compute Intended production MySQL hosting.
Google Cloud Redis Intended cache/session acceleration layer.
Akamai Load balancing, CDN and edge protection.

Coding Standards

- Use prepared statements for SQL.
- Validate roles and tenant ownership before data mutation.
- Escape all HTML output.
- Use CSRF protection on POST actions.
- Store uploads in private storage.
- Keep secrets in ‘.env‘, never in source code.
- Use clear function names and minimal inline comments.

Database Migrations

The current application performs runtime schema checks through helper functions. Production deployments should convert these schema changes into controlled migrations with versioning, rollback notes and staging verification.

API Endpoint Examples

Provider Ticket Update

Request:

‘‘‘http
POST /LMS/provider_api.php
Content-Type: multipart/form-data
‘‘‘

Payload:

```json
{
"action": "update_ticket",
"ticket_id": 15,
"status": "resolved",
"internal_note": "Issue resolved and customer notified.",
"_csrf": "session_csrf_token"
}
```

Response:

```json
{
"ok": true
}
```

Dashboard AJAX Action

Request:

```http
POST /LMS/admin_dashboard.php
```

Payload:

```json
{
"action": "save_course",
"section": "courses",
"title": "Mathematics Grade 10",
"description": "Core mathematics course",
"_csrf": "session_csrf_token",
"ajax": "1"
}
```

Response:

```json
{
"ok": true,
"message": "Course saved",
"flash_type": "success",
"redirect_url": "admin_dashboard.php?section=courses&flash=Course+saved&flash_type=success"
}
```

System Security

Data Encryption

Production environments must use TLS 1.2 or higher for all browser and API traffic. Database

connections should be encrypted where supported. Sensitive secrets must be stored outside the codebase.

Password Handling

Passwords are hashed using Argon2id. Imported users are forced to reset their default password on first login. Reset links use hashed tokens with expiry.

Access Control

Access control is role-based and tenant-scoped. Provider users have separate authentication from institution users.

Audit Logging

Production audit logging should include login attempts, provider actions, reset-link issuance, ticket changes, payment changes, user imports and privileged configuration updates.

POPIA Compliance Measures

The system supports purpose limitation, role-based access, data minimisation, access controls, retention planning, breach response procedures and data subject request handling.

Child Data Protection Measures

Learner records may include child personal information. Institutions must ensure lawful authority, guardian consent where required, age-appropriate processing, restricted access and safe communication practices.

Data Retention Strategy

Retention periods must be defined by contract, law and institutional policy. Recommended categories include active learning records, archived academic records, support records, payment logs, audit logs and deleted user records.

Backup and Disaster Recovery

Recommended measures include encrypted database backups, storage replication, restore testing, offsite backup copies, documented RPO/RTO targets and incident response runbooks.

References

- Information Regulator South Africa: POPIA and PAIA guidance.
- European Commission: GDPR principles and data subject rights.
- California Privacy Protection Agency and California Attorney General: CCPA/CPRA rights and obligations.
- European Commission and ENISA: NIS2 cybersecurity governance guidance.