**Learning Management System (LMS) Project Specifications Document**

**Project Title:**

Custom LMS Development with SCORM Integration

**Project Description:**

This project aims to develop a scalable Learning Management System (LMS) to manage SCORM-compliant courses and support roles such as Admin, Instructor, and Student. The LMS will feature course management, progress tracking, and user-friendly dashboards tailored to each role.

**Functional Requirements**

**1. User Management**

* **Roles:** Admin, Instructor, Student.
* **Features:**
  + Registration and login functionality.
  + Role-based permissions and access.
  + Password reset and profile management.
  + User dashboards tailored to roles.

**2. Course Management**

* **Course Features:**
  + Create, update, delete courses.
  + Upload SCORM-compliant content.
  + Manage course modules (e.g., videos, PDFs, quizzes).
* **Instructor Features:**
  + Assign courses to specific students.
  + Monitor student progress.
* **Student Features:**
  + Enroll in courses.
  + Access course materials.

**3. Progress Tracking**

* Track course/module completion for students.
* Display progress on dashboards.

**4. SCORM Support**

* Upload and render SCORM packages.
* Track SCORM runtime data (e.g., scores, completion status).

**5. Notifications**

* In-app and email notifications for course updates, deadlines, and assignments.

**6. Reporting**

* Generate reports for Admins and Instructors to view user activity, course performance, and progress.

**7. Accessibility and Responsiveness**

* Ensure compatibility across devices (desktop, tablet, and mobile).

**Non-Functional Requirements**

**1. Security**

* Secure authentication using Django’s built-in mechanisms.
* Role-based access control for sensitive data.
* Secure storage for SCORM files.

**2. Scalability**

* Support up to 300 users with the ability to scale for future growth.
* Optimize database queries and storage for SCORM packages.

**3. Performance**

* Fast page load times.
* Efficient handling of SCORM runtime data.

**4. Maintainability**

* Modular app structure for easy updates.
* Detailed documentation for developers.

**Technical Specifications**

**1. Technology Stack**

* **Backend Framework:** Django (Python).
* **Frontend Framework:** HTML, CSS, JavaScript (future consideration for React).
* **Database:** SQLite for development, PostgreSQL for production.
* **Authentication:** Django built-in authentication with CustomUser model.
* **SCORM Integration:** Open-source SCORM player or custom implementation.

**2. Hosting**

* **Initial Deployment:** NAS or local environment.
* **Scalable Hosting:** AWS Lightsail or DigitalOcean for production.

**3. Tools**

* **Version Control:** Git.
* **Development Environment:** Visual Studio Code.
* **Package Manager:** pip (Python).
* **Testing Tools:** Django’s built-in test framework.

**Development Plan**

**Phase 1: User Management (Week 1–2)**

* Set up CustomUser model with roles (Admin, Instructor, Student).
* Develop registration and login functionality.
* Create user dashboards.

**Phase 2: Course Management (Week 3–4)**

* Develop models for courses and modules.
* Implement views and templates for course creation and listing.

**Phase 3: SCORM Integration (Week 5)**

* Research open-source SCORM players.
* Implement SCORM upload and playback features.

**Phase 4: Progress Tracking and Reporting (Week 6)**

* Add progress tracking functionality.
* Create dashboards with performance metrics.

**Phase 5: Testing and Deployment (Week 7–8)**

* Conduct unit and integration tests.
* Deploy on NAS or cloud environment.

**Future Development Initiatives**

**Future AI Integration Points**

**Content Enhancement:**

* Automatic content summarization.
* Question generation from content.
* Content recommendation engine.

**Personalized Learning:**

* Learning path optimization.
* Difficulty adjustment.
* Study pattern analysis.

**Assessment Enhancement:**

* AI-powered grading.
* Question quality analysis.
* Cheating detection.

**Student Support:**

* AI tutoring assistant.
* Query resolution.
* Progress prediction.

**Immediate Next Steps**

* Complete role-based dashboards for Admin, Instructor, and Student.
* Implement course management functionality in the next development phase.

This document serves as a comprehensive reference for the LMS project, ensuring alignment across functional and technical requirements while providing a roadmap for development.

Updated with Data structures and additional features : 19/12/2024, 12:34

## Software Specification Document: Learning Management System (LMS)

### ****Project Overview****

The goal of this project is to develop a comprehensive, scalable Learning Management System (LMS) that incorporates advanced features like role-based access, course management, gamification, reporting, e-commerce, and integrations. The LMS will serve multiple roles (SuperAdmin, Admin, Instructor, Learner) and cater to various branches and groups.

### ****Functional Requirements****

#### ****1. User Management****

* **Roles**:
  + SuperAdmin: Full access to all branches and system settings.
  + Admin: Branch-specific access, including user and course management.
  + Instructor: Assigned course and learner management.
  + Learner: Access to enrolled courses and progress tracking.
* **Features**:
  + User creation and management (name, email, bio, role, timezone, language).
  + Group and branch assignment.
  + Password management and security settings.
  + Role-based access controls (RBAC).

#### ****2. Course Management****

* **Course Features**:
  + Create, update, and archive courses.
  + Support for categories and subcategories.
  + Prerequisites and dynamic learning paths.
  + Units (modules) including text, video, SCORM, and assessments.
* **Assessment Features**:
  + Tests, assignments, surveys.
  + Scoring and pass criteria.
  + Attempt limits.

#### ****3. Gamification****

* Points, badges, and levels for user engagement.
* Leaderboards at group and branch levels.
* Customizable rewards for milestones.

#### ****4. Reporting****

* Types of Reports:
  + User activity (logins, progress, completions).
  + Course metrics (completion rates, training time, ratings).
  + Branch and group performance.
  + Custom reports with flexible rulesets.
  + Training matrix for cross-referenced user-course alignment.
  + Timeline for event-based tracking.
* Export options (Excel, CSV).

#### ****5. E-commerce****

* Subscriptions for courses (monthly/yearly).
* Discounts, invoicing, and coupon management.
* Integration with payment gateways (e.g., Stripe, PayPal).

#### ****6. Integrations****

* Web conferencing tools (e.g., Zoom, MS Teams).
* CRM platforms (e.g., Salesforce, Zendesk).
* Automation tools (e.g., Zapier, Shopify).
* API support for custom integrations.

#### ****7. Security****

* Enforced password policies and lockout mechanisms.
* Domain-restricted registrations.
* Content protection (video watermarking, document restrictions).
* Single Sign-On (SSO).

#### ****8. Import/Export****

* Bulk import/export of user and course data.
* FTP support for automated data sync.

### ****Non-Functional Requirements****

#### ****1. Scalability****

* Support for large user bases across multiple branches.
* Optimized database queries for reports and progress tracking.

#### ****2. Performance****

* Real-time updates for dashboards and reports.
* Fast load times for pages and content.

#### ****3. Security****

* Secure authentication mechanisms.
* Data encryption for sensitive fields.

#### ****4. Maintainability****

* Modular architecture for easy updates.
* Comprehensive developer documentation.

#### ****5. Accessibility****

* Responsive design for desktop, tablet, and mobile.
* Compliance with accessibility standards (e.g., WCAG).

### ****Data Structures****

#### ****1. Users****

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| user\_id | Primary Key | Unique identifier for users. |
| username | String | User login name. |
| email | String | User email address. |
| role | Enum | Role (SuperAdmin, Admin, etc.) |
| branch\_id | Foreign Key | Linked branch. |
| language | String | User language preference. |
| timezone | String | User timezone. |
| failed\_attempts | Integer | Failed login attempts. |

#### ****2. Courses****

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| course\_id | Primary Key | Unique identifier for courses. |
| title | String | Course name. |
| category\_id | Foreign Key | Linked category. |
| availability | Enum | Open, Restricted, Scheduled. |
| price | Decimal | Price for paid courses. |
| prerequisites | JSON | Learning path prerequisites. |
| created\_by | Foreign Key | Creator user ID. |

#### ****3. Reports****

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| report\_id | Primary Key | Unique identifier for reports. |
| type | Enum | Report type (user, course). |
| created\_by | Foreign Key | Creator user ID. |
| filters | JSON | Filters for report generation. |
| output\_format | Enum | Excel, CSV. |

### ****Development Plan****

#### ****Phase 1: Core Features (Weeks 1-4)****

* User and role management.
* Course creation and unit management.
* Basic reports (users, courses).

#### ****Phase 2: Advanced Features (Weeks 5-8)****

* Gamification (points, badges, leaderboards).
* E-commerce features (subscriptions, coupons).
* Security enhancements.

#### ****Phase 3: Scalability (Post Launch)****

* Import/export functionality.
* Integration support (web conferencing, CRM).
* Automation tools (Zapier, custom APIs).