'''

Lixnet School Management System - System Requirements Specification (SRS)

Document Version: 2.0

Prepared by: Joshua Otieno

Date: 2025-09-19

Table of Contents

1. Introduction

This System Requirements Specification (SRS) document provides a comprehensive description of the Lixnet School Management System (SMS). It outlines the system's purpose, scope, functional and non-functional requirements, and serves as a foundational agreement between the development team and the stakeholders. This document is intended to guide the design, development, testing, and deployment of the Lixnet SMS, ensuring that the final product meets the specified requirements and expectations.

1.1. Purpose

The purpose of this document is to provide a detailed and unambiguous specification of the Lixnet School Management System. It is intended for a wide audience, including:

- **Project Managers:** To understand the project scope and plan the development process.
- **Developers:** To understand what to build and how the system should behave.
- **QA and Testers:** To create test cases and ensure the system meets the specified requirements.
- **Stakeholders:** To have a clear understanding of the system's capabilities and to provide feedback.

1.2. Scope and Objectives

Scope: The Lixnet School Management System is a modular, scalable, and secure web-based application designed to automate and streamline the administrative and academic processes of educational institutions. The system will be accessible to various user roles, including administrators, teachers, students, parents, and finance/support staff. The core functionalities of the system include user management, student and staff information management, course and timetable management, attendance tracking, examination and grading, financial management with M-Pesa integration, communication and notifications, and an optional library management module.

Key Objectives:

- To replace manual and fragmented processes with a single, unified platform.
- To provide role-based access control (RBAC) and support for multiple schools (multi-tenancy).
- To deliver the system in modular increments with a continuous integration and delivery (CI/CD) approach.

- To provide comprehensive documentation and APIs for future integrations and extensions.

2. Overall Description

2.1. Product Perspective

The Lixnet School Management System is a modular monolith, API-driven web application. The frontend will be built using React, which will communicate with a Laravel-based backend through a RESTful JSON API. The system's modules are designed to be independent yet integrated, sharing a common authentication and data model.

2.2. User Roles and Permissions

The system will support the following user roles with their respective permissions:

Role	Permissions
Super Admin	Full system access, including managing schools and system-wide settings.
School Admin	Manages students, staff, finances, and reports within their assigned school.
Teacher	Manages class assignments, attendance, exams, and grades for their assigned classes and subjects.
Student	Views their timetable, attendance records, grades, and can pay fees.
Parent	Views their child's data, receives notifications, and can make payments.
Finance Officer	Manages invoices, receives payments, and performs financial reconciliation.
Librarian	Manages the library's inventory and transactions (if the library module is enabled).

2.3. Operating Environment

- **Frontend:** A responsive React application, hosted on a Content Delivery Network (CDN) or static hosting service behind a reverse proxy.
- **Backend:** A PHP (Laravel) application providing RESTful APIs with an OpenAPI specification.

- **Database:** MySQL or MariaDB.
- **Deployment:** The system will be deployed using Docker containers in both staging and production environments.

3. Functional Requirements

This section details the functional requirements of the Lixnet School Management System, categorized by module. Each requirement is assigned a unique identifier (FR-###) for traceability.

3.1. Module 1: User Management & Authentication

ID	Requirement
FR-101	Role-based authentication and authorization (RBAC).
FR-102	User CRUD for Admins (create users, set roles).
FR-103	Login, logout, password reset via email.
FR-104	Optional MFA (TOTP) for Admin & Finance users.
FR-105	Session management and inactivity timeout.
FR-106	Audit logs for login attempts and admin actions.

3.2. Module 2: Student Information System (SIS)

ID	Requirement
FR-201	Student profile CRUD (bio, guardian contacts, medical, enrollment history).
FR-202	Enrollment workflow (new admission, transfer in/out).
FR-203	Class allocation and promotion history.
FR-204	Import/export CSV for bulk student data.

3.3. Module 3: Staff & Teacher Management

ID	Requirement

FR-301	Staff profiles (roles, subjects, schedule).
FR-302	Staff assignment to classes and subjects.
FR-303	Staff leave request and approval workflow (optional).

3.4. Module 4: Course & Class Management

ID	Requirement
FR-401	Define subjects/courses, class groups, academic terms.
FR-402	Timetable creation and conflict checks.
FR-403	Classroom and resource allocation.

3.5. Module 5: Attendance Tracking

ID	Requirement
FR-501	Student daily attendance marking (manual and bulk).
FR-502	Teacher attendance tracking.
FR-503	Attendance reports by class, student, date-range.
FR-504	Trigger notifications for persistent absences.

3.6. Module 6: Exams & Grading

ID	Requirement
FR-601	Create exam templates and schedules.
FR-602	Marks entry UI (teacher) and bulk upload.
FR-603	Grade calculation rules (weighting, grade boundaries).
FR-604	Generate report cards and transcripts (PDF export).
FR-605	Exam analytics (class averages, top performers).

3.7. Module 7: Finance & Fees Management (M-Pesa)

ID	Requirement
FR-701	Fees structure per class/term and concessions.
FR-702	Generate invoices and receipts.
FR-703	Integrate with M-Pesa APIs for payments (STK Push & Reconciliation).
FR-704	Payment reconciliation dashboard; manual adjustments.
FR-705	Exportable financial reports (CSV, PDF).

3.8. Module 8: Communication & Notifications

ID	Requirement
FR-801	Announcements (global and class-level).
FR-802	Email and SMS notifications via providers (templated).
FR-803	Push notifications (in-app) and digest emails.
FR-804	Opt-in/opt-out settings for parents/students.

3.9. Module 9: Library Management (Optional)

ID	Requirement
FR-901	Inventory of books/resources, issue/return tracking.
FR-902	Fine calculation rules and integration with Finance module.

3.10. Module 10: Analytics & Reporting

ID	Requirement
FR-1001	Dashboard widgets (attendance, fees collected, exam performance).
FR-1002	Custom report builder and scheduled report exports.

FR-1003	Role-based visibility for reports.

4. External Interface Requirements

This section specifies the external interfaces of the Lixnet School Management System.

4.1. User Interfaces

The primary user interface will be a responsive web application accessible through modern web browsers. The UI will be designed to be intuitive and user-friendly, with a focus on providing a seamless experience across different devices.

4.2. Hardware Interfaces

There are no specific hardware interfaces required for the Lixnet SMS.

4.3. Software Interfaces

- **M-Pesa API:** The system will integrate with the Safaricom M-Pesa API for processing fee payments. This integration must comply with all the security and communication protocols specified by Safaricom.
- **SMS/Email Gateway:** The system will use a third-party SMS/Email gateway for sending notifications to users. The specific provider will be determined during the implementation phase.
- **Analytics Tools:** The system may integrate with third-party analytics tools for generating reports and dashboards. The selection of these tools will be based on the specific reporting requirements.

5. Non-Functional Requirements

This section defines the non-functional requirements (NFRs) of the Lixnet School Management System. These requirements specify the quality attributes of the system and are essential for ensuring a good user experience and a robust and reliable system.

ID	Category	Requirement
NFR-1	**Security**	The system must mitigate the OWASP Top 10 vulnerabilities. All passwords must be hashed using a strong algorithm (e.g., bcrypt or argon2). All communication must be encrypted using TLS. API access

		must be secured using tokens, and the principle of least privilege must be enforced through role-based access control.
NFR-2	**Performance**	The system must support up to 1,000 concurrent users for a single-school deployment. API response times for typical endpoints should be less than 300ms.
NFR-3	**Scalability**	The backend architecture must support horizontal scaling behind a load balancer. The database design should allow for future scaling through techniques like sharding or read-replicas.
NFR-4	**Availability**	The system must have an uptime target of 99.5%. Scheduled maintenance windows will be documented and communicated to users in advance.
NFR-5	**Maintainability**	The codebase must be clean, modular, and adhere to established coding standards. The system must have a comprehensive suite of automated tests, including unit and integration tests.
NFR-6	**Internationalization**	The system must support multiple languages and date formats. Currency formatting should be based on the user's locale.
NFR-7	**Accessibility**	The core pages of the application must comply with the Web Content Accessibility Guidelines (WCAG) 2.1 Level AA.
NFR-8	**Data Retention and Backups**	The system must perform daily database backups with a minimum of 30-day restore points.

6. Data Model

This section provides a high-level overview of the data model for the Lixnet School Management System. A detailed Entity-Relationship (ER) diagram will be provided during the design phase.

6.1. High-Level Entities

The following are the high-level entities in the system:

- **Users:** Stores information about all users of the system, including their roles and credentials.
- **Roles:** Defines the different user roles in the system and their associated permissions.
- **Students:** Stores detailed information about each student, including their personal details, guardian information, and enrollment history.
- **Guardians:** Stores information about the parents or guardians of the students.
- **Staff:** Stores information about the teaching and non-teaching staff of the school.
- **Classes:** Represents the different classes or grades in the school.
- **Subjects:** Stores information about the subjects taught in the school.
- **Timetables:** Stores the class schedules for different classes and teachers.
- **AttendanceRecords:** Stores the attendance records of students.
- **Exams:** Stores information about the examinations conducted in the school.
- **Marks:** Stores the marks obtained by students in different exams.
- **Invoices:** Stores the fee invoices generated for students.
- **Payments:** Stores the payment records for the invoices.
- **Notifications:** Stores the notifications sent to users.
- **LibraryItems:** Stores information about the books and other resources in the library.

6.2. System Context Diagram

(A system context diagram will be inserted here to visually represent the system's boundaries and its interactions with external entities.)

6.3. Entity-Relationship Diagram

(A detailed ER diagram will be inserted here during the design phase to illustrate the relationships between the different entities in the system.)

7. API Design Principles

This section outlines the principles that will guide the design of the RESTful API for the Lixnet School Management System.

- **RESTful Architecture:** The API will adhere to the principles of REST (Representational State Transfer), using standard HTTP methods (GET, POST, PUT, DELETE) for resource manipulation.
- **JSON-Based:** All API responses will be in JSON format.

- **Versioning:** The API will be versioned to ensure backward compatibility. The version will be included in the API endpoint (e.g., `/api/v1/...`).
- **Authentication:** API access will be secured using JSON Web Tokens (JWT) for API clients and session tokens for the web UI.
- **Consistent Naming:** The API endpoints and resource names will follow a consistent and predictable naming convention.
- **OpenAPI Specification:** A comprehensive OpenAPI (Swagger) specification will be generated for all public endpoints, providing detailed documentation for API consumers.

8. UI/UX Requirements

This section outlines the user interface (UI) and user experience (UX) requirements for the Lixnet School Management System.

- **Responsive Design:** The application must be responsive and provide an optimal viewing experience across a wide range of devices, from desktops to mobile phones. The design should be mobile-first, but with a focus on a great desktop experience as well.
- **Component Library:** A reusable component library will be developed to ensure a consistent look and feel throughout the application. This library will include standard UI elements such as buttons, forms, tables, and modals.
- **Role-Specific Dashboards:** Each user role will have a dedicated dashboard that provides quick access to the most relevant information and actions.
- **Clear Error Handling:** The application must provide clear and concise error messages and validation feedback to guide users in correcting their inputs.
- **Intuitive Navigation:** The navigation structure of the application should be intuitive and easy to use, allowing users to find the information they need with minimal effort.

9. Workflows & User Stories

This section provides a set of user stories to illustrate the expected user interactions with the Lixnet School Management System. These user stories are not exhaustive but are intended to provide a clear understanding of the system's functionality from a user's perspective.

9.1. New Student Enrollment

- **As a School Admin,** I want to create a new student profile, upload necessary documents (e.g., birth certificate, previous school records), and assign the student to a class, so that the student's information is accurately recorded in the system and they are ready to start their classes.
- **Acceptance Criteria:**
  - The new student appears in the assigned class list.
  - The student's fee structure is automatically assigned based on their class.
  - The student's parents or guardians receive an automated welcome email with instructions on how to access the parent portal.

9.2. Teacher Enters Marks

- **As a Teacher,** I want to enter the marks for my subject for a specific exam, so that the students' report cards can be generated accurately and on time.
- **Acceptance Criteria:**
  - The marks are saved successfully in the system.
  - The corresponding grades are calculated automatically based on the predefined grading rules.
  - The marks and grades are visible to the student and their parents on their respective portals.

9.3. Parent Pays Fees

- **As a Parent,** I want to be able to view my child's outstanding fee invoices and pay them securely online using M-Pesa, so that I can easily manage my financial obligations to the school.
- **Acceptance Criteria:**
  - The payment status is updated automatically in the system upon successful payment.
  - A receipt is generated and sent to my registered email address.
  - The school's finance ledger is updated to reflect the payment.

10. Testing & QA Plan

This section outlines the testing and quality assurance (QA) plan for the Lixnet School Management System. The goal of this plan is to ensure that the system is of high quality, free of defects, and meets the specified requirements.

10.1. Testing Levels

- **Unit Tests:** Each individual component of the backend (PHP) and frontend (React) will be tested in isolation to ensure that it functions correctly.
- **Integration Tests:** Critical workflows, such as user authentication, payment processing, and exam grading, will be tested to ensure that the different components of the system work together as expected.
- **End-to-End (E2E) Tests:** The major user journeys will be tested from start to finish to simulate real-world usage and identify any issues that may arise in a production environment.

10.2. Testing Types

- **Functional Testing:** The system will be tested to ensure that it meets all the functional requirements specified in this document.
- **Regression Testing:** After each new feature is added or a bug is fixed, the system will be re-tested to ensure that the changes have not introduced any new issues.
- **Performance Testing:** The system will be tested to ensure that it meets the performance requirements specified in the non-functional requirements section.
- **Security Testing:** The system will be tested for security vulnerabilities to ensure that it is protected against common threats.

10.3. Test Environment

A dedicated staging environment will be set up for testing purposes. This environment will be a replica of the production environment and will be populated with realistic test data.

11. Deployment & CI/CD

This section outlines the deployment and Continuous Integration/Continuous Deployment (CI/CD) strategy for the Lixnet School Management System.

- **CI/CD Pipeline:** A CI/CD pipeline will be set up using GitHub Actions. This pipeline will automatically run tests, perform linting, and build the application artifacts whenever new code is pushed to the repository.
- **Staging and Production Environments:** The application will be deployed to a staging environment for testing and quality assurance before being promoted to the production environment. The promotion to production will be a manual step to ensure that only stable and tested code is released to users.

- **Dockerized Services:** All the services of the application will be containerized using Docker. This will ensure consistency across different environments and simplify the deployment process.
- **Environment Configuration:** All environment-specific configurations will be managed using a secrets manager to ensure that sensitive information is not hard-coded in the application.

12. Monitoring & Logging

This section outlines the monitoring and logging strategy for the Lixnet School Management System.

- **Centralized Logging:** All application and system logs will be centralized in a logging service (e.g., ELK stack or a hosted service). This will make it easier to search, analyze, and troubleshoot issues.
- **Performance Monitoring:** An Application Performance Monitoring (APM) tool will be used to monitor the performance of the application, including response times, error rates, and resource utilization.
- **Alerting:** Alerts will be configured to notify the development team via Slack or Teams in case of critical failures or performance degradation.

13. Security & Compliance

This section outlines the security and compliance requirements for the Lixnet School Management System.

- **TLS Encryption:** All traffic between the client and the server must be encrypted using TLS (Transport Layer Security).
- **Role-Based Access Control:** The system must enforce the principle of least privilege through a robust role-based access control (RBAC) mechanism.
- **Input Validation and Sanitization:** All user inputs must be validated and sanitized to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS).
- **Vulnerability Scanning:** The system will undergo regular vulnerability scans and dependency audits to identify and address potential security risks.
- **Data Protection:** The system must comply with all local data protection regulations. It must also provide processes for data export and erasure to fulfill data subject requests.

14. Backup & Recovery

This section outlines the backup and recovery plan for the Lixnet School Management System.

- **Automated Backups:** The system will have automated nightly backups of the database.
- **Point-in-Time Recovery:** The system will support point-in-time recovery, with a minimum of 30-day restore points.
- **Disaster Recovery Plan:** A disaster recovery plan will be documented, outlining the steps to be taken in case of a major system failure.

15. Documentation Deliverables

This section lists the documentation that will be delivered as part of the Lixnet School Management System project.

- **Technical Documentation:**
  - Architecture Document
  - Entity-Relationship (ER) Diagrams
  - API Specification (OpenAPI)
  - Deployment Guides
- **User Documentation:**
  - Admin Guide
  - Teacher Guide
  - Parent/Student Guide
  - Frequently Asked Questions (FAQ)
- **Onboarding Documentation:**
  - Onboarding Checklist for New Schools

16. Roles, Responsibilities & Team Assignment

This section outlines the roles, responsibilities, and team assignments for the development of the Lixnet School Management System. The team consists of five developers and a team lead.

- **Team Lead:** Project coordination, GitHub governance, sprint planning, stakeholder communication, and hands-on technical alignment.
- **Dev 1 (Frontend Owner):** React components, UI, responsive layouts, storybook.
- **Dev 2 (Backend Owner):** Laravel APIs, auth, core business logic.

- **Dev 3 (DB & Integrations):** Schema design, migrations, M-Pesa & SMS/Email provider integration.
- **Dev 4 (QA & Integration):** Test suites, E2E, QA sign-off, integration testing.
- **Dev 5 (Docs & DevOps):** CI/CD, Docker, hosting, runbooks, technical documentation.

17. Roadmap & Sprint Plan

This section outlines the proposed 10-week roadmap and sprint plan for the development of the Lixnet School Management System.

Week	Sprint Goal
Week 0	**Preparation:** Repository setup, CI/CD configuration, task board creation, and team onboarding.
Week 1	**Module 1:** User Management & Authentication (FR-101 to FR-106).
Week 2	**Module 2:** Student Information System (SIS).
Week 3	**Module 3:** Staff & Teacher Management.
Week 4	**Module 4:** Course & Class Management.
Week 5	**Module 5:** Attendance Tracking.
Week 6	**Module 6:** Exams & Grading.
Week 7	**Module 7:** Finance & Fees (M-Pesa integration).
Week 8	**Module 8:** Communication & Notifications.
Week 9	**Module 9:** Library Management (optional) & Polish.
Week 10	**Module 10:** Analytics, Reporting, Final QA & Handover.

Each sprint will follow a one-week cadence, including planning, development, review, testing, demo, and retrospective.

18. Acceptance Criteria & Sign-off

This section defines the acceptance criteria for the Lixnet School Management System.

- **Module-Level Acceptance:** Each module must have a feature checklist, and all unit and integration tests must pass. The module must be deployed to the staging environment and demonstrated to the stakeholders. A user guide for the module must also be completed.
- **Final Sign-off:** The final sign-off will be provided by Lixnet management after the User Acceptance Testing (UAT) is completed and all major issues have been resolved.

19. Risks & Mitigations

This section identifies potential risks that could impact the Lixnet School Management System project and outlines mitigation strategies to address them.

Risk	Mitigation
M-Pesa Integration Delays	Begin the integration process early in the development cycle. Develop a prototype with the M-Pesa sandbox environment to identify potential challenges. Have a fallback plan for manual payment reconciliation in case of unexpected delays.
Team Skill Gaps	Encourage pair programming and knowledge sharing sessions. Rotate tasks among team members to build cross-functional skills. Allocate a buffer sprint in the project plan to address any unforeseen challenges.
Scope Creep	Freeze the Minimum Viable Product (MVP) scope at the beginning of the project. Create a backlog for all new feature requests and prioritize them for future phases of the project.

20. References

[1] Perforce. (2025, July 11). *How to Write a Software Requirements Specification (SRS) Document*. Perforce. https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

21. Appendix

This section contains supplementary materials that are not essential to the main body of the SRS but provide additional context and information.

- **Templates:**
  - Stand-up Meeting Template

    - Pull Request (PR) Checklist
    - QA Checklist
    - Release Notes Template
- **Sample Data:**
    - Sample API Contract Examples
    - Sample CSV Import Templates
    '''