EEC

Electronic Educare

Software Requirements Specification

Multi-Tenant School Management System

Document Version:	1.0
Status:	Draft
Date:	2026-01-12
Maintainers:	EEC Product & Architecture Team
Classification:	Confidential

Document Version: 1.0

Date: 2026-01-12

Maintainers: EEC Product & Architecture Team

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) consolidates the behavioral and non-behavioral requirements for the EEC multi-tenant ERP/LMS platform. It translates the capabilities implemented in the current codebase ('backend/', 'frontend/') and supporting documentation ('README.md', 'docs/*') into structured requirements that guide product planning, implementation, testing, and onboarding of new contributors.

1.2 Scope

- Product Name:: EEC Multi-Tenant School Management System
- Deployment Model:: Multi-tenant SaaS serving multiple schools and campuses with shared infrastructure and tenant isolation enforced via 'schoolId'.
- Primary Users:: Institution heads, school admins, principals, teachers, class teachers, students, parents/guardians, finance/accounting staff, support staff.
- Primary Modules:: Authentication, tenant onboarding, academic structure setup, user management, student lifecycle (including NIF-specific workflows), attendance, assignments/exams, AI-powered learning analytics, fee & finance, communications/notices, dashboards/reports, bulk data operations, wellbeing/support services.

1.3 Definitions, Acronyms, Abbreviations

- EEC:: Educational Excellence Center (organization).

- Tenant:: Logical school entity identified by 'schoolId'.
- RBAC:: Role-Based Access Control.
- AI Learning:: Module that analyzes student weaknesses and prescribes learning paths.
- ERP/LMS:: Enterprise Resource Planning / Learning Management System.
- API:: RESTful HTTP endpoints served by Express.js backend.
- KPI:: Key Performance Indicator.

1.4 References

- Project overview and role matrix ('README.md').
- System, data-flow, and access documentation ('docs/system-diagram.md', 'docs/data-flow.md', 'docs/access-matrix.md').
- Architecture suite ('docs/architecture/*.md').
- API catalog ('docs/api-reference.md').
- Bulk upload quick reference ('QUICK-REFERENCE.txt').
- Frontend experience summary ('frontend/features.txt').
- Backend route implementations ('backend/routes/*.js').

1.5 Document Overview

Section 2 outlines the product context and constraints. Section 3 details functional requirements grouped by system features. Section 4 captures external interface expectations. Section 5 enumerates non-functional requirements. Appendices offer glossary and traceability notes.

2. Overall Description

2.1 Product Perspective

- React SPA frontend (Vite build, Tailwind CSS) communicates with an Express 5 API ('backend/index.js') secured by JWT, tenant resolver middleware, and RBAC.
- MongoDB Atlas stores multi-tenant data; Cloudinary manages media assets; optional Redis caching is planned.
- External email/SMS gateways are used for notifications.
- System exposes >100 REST endpoints covering admin, academic, finance, AI, reporting, and student-engagement functions.

2.2 Product Functions

1. Provide multi-role authenticated portals (student, parent, teacher, admin, principal). 2. Allow institution heads to onboard schools and configure academic hierarchies, classes, sections, subjects, and timetables. 3. Manage user lifecycle (creation, assignment, deactivation) for staff, students, and guardians including bulk onboarding. 4. Track academic activities: attendance, assignments, exams, lesson plans, behavior, progress, wellbeing, and AI-driven insights. 5. Administer financial operations: fee structures, invoicing, payments, and NIF-specific course fees. 6. Offer dashboards, analytics, and reports tailored per role (e.g., principal overview, admin summary). 7. Facilitate communication (notices,

messages, alcove Q&A, notifications). 8. Support CSV/XLS student imports, audit logging, and attachments via upload endpoints.

2.3 User Classes and Characteristics

- Super Admin / Institution Head:: Manages multi-school provisioning, requires full CRUD and oversight.
- School Admin:: Operates within school/campus boundary; responsible for academic setup, staff, students, finance, reports.
- Principal:: Consumes analytical dashboards, approves schedules, monitors KPIs.
- Teacher / Class Teacher:: Delivers instruction, records attendance, assignments, grades, lesson plans, meets parents.
- Student:: Consumes personal dashboard, submits work, views attendance/results, interacts with AI learning aids.
- Parent/Guardian:: Views child progress, pays fees, schedules meetings, receives alerts.
- Accountant:: Manages fee structures, invoices, payments, financial reports.
- Support/Wellbeing Staff:: Handles cases from support, wellbeing, behavior, and progress trackers.
- System Integrations:: External notification services, Cloudinary storage.

2.4 Operating Environment

- Frontend:: React 18 SPA served via Vercel/S3/Static host; modern browsers (Chrome, Edge, Firefox, Safari) on desktop/mobile; responsive layouts.
- Backend:: Node.js 18+ runtime running Express 5 API behind Nginx/load balancer; deployed on Render/EC2; uses dotenv-managed secrets.
- Database:: MongoDB Atlas clusters (M10–M30).
- Storage:: Cloudinary CDN for media uploads; local 'uploads/' for development.
- CI/CD:: Vercel for frontend preview, Node server for backend; Docker/Kubernetes supported per architecture doc.
- Tools:: Git, nodemon, ESLint, Swagger for API documentation.

2.5 Design and Implementation Constraints

- Multi-tenant isolation enforced via 'schoolId' on every collection; middleware must not be bypassed.
- JWT tokens required in all protected routes; admin, teacher, student, and parent tokens carry role & tenant claims.
- Strong password hashing (bcrypt) and rate limiting (express-rate-limit).
- Frontend uses Tailwind CSS 4.1 and Vite 6.3; backward compatibility with React 18 until upgrade plan finalizes.

- File uploads rely on Multer configuration; Cloudinary credentials must be provided.
- Network access to email/SMS providers may be restricted per deployment environment.
- Must comply with data retention and audit logging strategies defined in 'docs/architecture/02-database-architecture.md'.

2.6 Assumptions and Dependencies

- Schools operate academic years with class-section hierarchies; students belong to exactly one class/section per year.
- Guardians can be linked to multiple students but every student has at least one guardian contact.
- Fee structures vary by course/grade; invoice generation depends on structure availability.
- AI learning module requires baseline assessment data per subject to generate learning paths.
- Bulk upload files conform to templates described in 'QUICK-REFERENCE.txt' and validated server-side.
- External communication (email/SMS) and AI analytics rely on configured third-party credentials/services.
- Future enhancements include Redis caching, microservices extraction, and monitoring (Prometheus/Sentry) per roadmap.

3. Functional Requirements

Requirements are organized by feature set. Each item lists priority (H/M/L) and acceptance criteria cues.

3.1 Authentication & Access Control

Requirement ID	Pri	Description
FR-AUTH-01	H	System shall provide registration and login endpoints per role ('/api/{role}/auth/*', 'docs/api-reference.md'). Successful login returns JWT embedding 'role', 'userId', 'schoolId'.
FR-AUTH-02	H	System shall enforce RBAC middleware ensuring role-based route access ('backend/middleware/*'). Teachers/students/parents may only access scoped resources.
FR-AUTH-03	H	System shall apply rate limiting on authentication endpoints (10 requests/minute) to mitigate brute-force attempts.

Requirement ID	Pri	Description
FR-AUTH-04	H	Passwords shall be hashed using bcrypt (10 rounds).
FR-AUTH-05	M	System shall support token refresh by re-authenticating; tokens expire in one day.
FR-AUTH-06	M	Audit logs shall record privileged authentication actions ('backend/routes/auditLogRoutes.js').

3.2 Tenant & Institution Management

Requirement ID	Pri	Description
FR-TEN-01	H	Super Admin shall create schools via '/api/schools' (POST) capturing name, domain metadata, contact, campus info.
FR-TEN-02	H	Super Admin shall list/retrieve schools for oversight ('GET /api/schools', '/api/schools/:id').
FR-TEN-03	H	Every tenant request shall resolve 'schoolId' before hitting controllers; unauthorized or invalid IDs must return HTTP 400/401 as shown in 'backend/routes/nifStudentRoutes.js'.
FR-TEN-04	M	System shall support multiple campuses per school with campus-level IDs when available.
FR-TEN-05	M	School activation/deactivation states shall govern login access and background jobs.
FR-TEN-06	L	System shall expose school registration routes ('backend/routes/schoolRegistrationRoutes.js') for self-service onboarding with admin approval workflow.

3.3 User & Role Management

Requirement ID	Pri	Description
FR-USER-01	H	School Admin shall manage staff, student, parent creation via '/api/admin/users/create-user' with role selection and base profile data.

Requirement ID	Pri	Description
FR-USER-02	H	System shall support bulk user creation/import for students and parents via CSV/XLS uploads ('/api/admin/users/bulk-create-users', '/api/admin/users/bulk-import-csv', 'QUICK-REFERENCE.txt').
FR-USER-03	H	System shall auto-generate usernames/passwords for students/parents with prefixes derived from admin username and admission year (see 'backend/routes/nifStudentRoutes.js').
FR-USER-04	M	Guardian accounts shall link to associated students through relational fields to support parent dashboards.
FR-USER-05	M	Dashboard stats endpoint ('GET /api/admin/users/dashboard-stats') shall provide counts by role, statuses, and recent additions.
FR-USER-06	M	System shall allow profile updates for students ('/api/student/profile/update') with validation of contact and academic fields.
FR-USER-07	L	Deactivation/archival endpoints shall exist for alumni and staff turnover (e.g., 'backend/routes/nifStudentRoutes.js' archived list).
FR-USER-08	L	Parent-teacher meeting scheduling shall be handled through meeting routes ('backend/routes/meetingRoute.js') capturing agendas, time slots, and participants.

3.4 Academic Structure Management

Requirement ID	Pri	Description
FR-ACAD-01	H	School Admin shall configure academic years, classes, sections, and subjects using '/api/academic/*' endpoints before operations commence.
FR-ACAD-02	H	Class-section hierarchy must support unique combinations per academic year and be referenced by timetables, attendance, and assessments.
FR-ACAD-03	H	Subjects may be assigned to classes/sections and mapped to teachers via teacher allocation routes.

Requirement ID	Pri	Description
FR-ACAD-04	M	Timetable endpoints ('/api/timetable') shall support CRUD for period scheduling with teacher/room references.
FR-ACAD-05	M	Lesson plan routes shall let teachers upload lesson outlines, resources, and objectives tied to subjects/periods.
FR-ACAD-06	L	System shall persist elective subjects and flexible structures for institutes like NIF with course codes and durations.

3.5 Student Lifecycle & NIF Module

Requirement ID	Pri	Description
FR-NIF-01	H	NIF student routes shall manage specialized enrollment data: batch code, course, duration, serial numbers, guardian info, etc.
FR-NIF-02	H	Student codes/roll numbers must be unique; duplicates reject import requests with descriptive errors.
FR-NIF-03	M	Module shall support multi-campus admissions with prefixing strategy described in 'nifStudentRoutes.js'.
FR-NIF-04	M	Admins shall view archived/withdrawn students separately for compliance.
FR-NIF-05	L	Module shall sync with NIF course catalog maintained through '/api/nif/course/*' endpoints.

3.6 Attendance Management

Requirement ID	Pri	Description
FR-ATT-01	H	Teachers shall mark attendance per class/section/date via 'POST /api/attendance/mark' capturing presence/absence and optional remarks.
FR-ATT-02	H	Attendance listings shall be filterable by student/class and role (teacher vs admin) via '/api/attendance/all' and '/api/attendance/admin/all'.

Requirement ID	Pri	Description
FR-ATT-03	M	Students and parents shall have read-only access to their attendance history via portal dashboards.
FR-ATT-04	M	Excuse letter routes shall allow students/parents to submit absence justifications and allow admin decisions.
FR-ATT-05	L	Attendance data shall feed analytics (e.g., risk detection) for principal dashboards and AI modules.

3.7 Assignments, Exams, and Assessments

Requirement ID	Pri	Description
FR-ASM-01	H	Teachers shall create assignments ('POST /api/assignment/add' or '/api/assignment') with instructions, due dates, targeted classes.
FR-ASM-02	H	Students shall view assigned work via '/api/assignment/fetch' and submit deliverables (file upload if configured).
FR-ASM-03	H	Exam module shall provide schedule creation ('POST /api/exam/add'), mark entry, and publishing of results accessible to students/parents.
FR-ASM-04	M	Progress routes shall track continuous assessment metrics and generate per-student dashboards.
FR-ASM-05	M	Behavior and wellbeing routes shall record incidents or support tickets linked to students for counselor workflows.
FR-ASM-06	M	Practice routes shall host quizzes or practice exercises; system shall record attempts and provide analytics.
FR-ASM-07	L	Lesson plan, curriculum pacing, and academic calendar events shall integrate with assignments/exams for visibility.

3.8 AI-Powered Learning Analytics

Requirement ID	Pri	Description

Requirement ID	Pri	Description
FR-AI-01	H	Authorized staff shall trigger weakness analysis via '/api/ai-learning/analyze-weakness/:studentId' to determine focus areas.
FR-AI-02	H	System shall maintain AI-generated learning paths accessible via GET endpoints per student/subject.
FR-AI-03	M	Teachers/students shall update progress metrics for AI recommendations via PUT endpoints.
FR-AI-04	M	Identify weak students ('GET /api/ai-learning/weak-students') for targeted interventions displayed on dashboards.
FR-AI-05	L	AI module shall integrate with student portal (e.g., motivational tips, dynamic guidance) per frontend design.
FR-AI-06	L	Future requirement: plug-in architecture to swap AI models without API contract changes.

3.9 Fee & Finance Management

Requirement ID	Pri	Description
FR-FEE-01	H	Fee structure endpoints shall allow admin/accountant to define course/grade fees, components, due dates ('/api/fees/structures').
FR-FEE-02	H	System shall generate invoices per student/term with statuses (pending, paid, overdue) accessible through '/api/fees/invoices'.
FR-FEE-03	H	Payment endpoint shall record transactions, mode, references, and optionally integrate with payment gateways.
FR-FEE-04	M	Parents/students shall view outstanding dues and payment history in portal; principal dashboard shows aggregated revenue.
FR-FEE-05	M	Fee module shall support NIF course fees and scholarships/discounts.

Requirement ID	Pri	Description
FR-FEE-06	L	Exportable financial reports (CSV/PDF via 'jspdf' and Chart.js components) shall be available for admins/principals.
FR-FEE-07	L	Audit logs shall capture fee edits and deletions.

3.10 Communication & Engagement

Requirement ID	Pri	Description
FR-COM-01	H	Notifications endpoints allow admins/teachers to broadcast notices, targeted alerts, and track user-specific inbox ('/api/notifications*').
FR-COM-02	H	Alcove community routes provide posting, editing, and commenting for Q&A/discussion.
FR-COM-03	M	Support routes shall intake helpdesk issues and assign statuses/resolution notes.
FR-COM-04	M	Issue/feedback routes allow students/parents to submit feedback; staff respond via dashboard.
FR-COM-05	M	Meeting routes coordinate parent-teacher or staff meetings with scheduling and reminders.
FR-COM-06	L	Notification system integrates with email/SMS services; fallback to in-app alerts when providers fail.
FR-COM-07	L	Principal quick actions hub triggers emergency alerts and broadcast messages with severity levels.

3.11 Reports & Analytics

Requirement ID	Pri	Description
FR-REP-01	H	'/api/reports/summary' shall return aggregated KPIs (student counts, attendance %, fee collection) for admin dashboards.
FR-REP-02	H	'/api/principal/overview' shall provide principal-level metrics: academics, finance, infrastructure, system health (see 'frontend/features.txt').

Requirement ID	Pri	Description
FR-REP-03	M	Teacher dashboards shall show course performance, assignment completions, and meeting schedules.
FR-REP-04	M	Reports must be filterable by period, class, course, campus; exported via PDF/CSV.
FR-REP-05	L	Predictive analytics highlighting at-risk students shall leverage AI module outputs.
FR-REP-06	L	Support KPI dashboards covering support tickets, wellbeing cases, attendance anomalies.

3.12 Bulk Operations & File Management

Requirement ID	Pri	Description
FR-BULK-01	H	CSV/XLS upload endpoints shall validate headers ('name,mobile,gender,...') and required columns before processing (see 'student-bulk-upload-template.csv').
FR-BULK-02	H	Bulk import must be transactional per row: successes recorded, failures returned in response with row index and reason.
FR-BULK-03	M	Upload limits (file size, row count) shall be enforced to prevent resource exhaustion.
FR-BULK-04	M	Upload routes store files temporarily under 'uploads/' and optionally push to Cloudinary; cleanup is required after processing.
FR-BULK-05	L	Export endpoints shall provide templates and actual data dumps for offline analysis.

3.13 Support, Wellbeing, and Behavior Modules

Requirement ID	Pri	Description
FR-SUP-01	M	Support routes allow creation, assignment, status tracking of support tickets per student.
FR-SUP-02	M	Wellbeing routes capture counselor notes, recommendations, and confidentiality flags.

Requirement ID	Pri	Description
FR-SUP-03	M	Behavior routes document incidents, consequences, and follow-up actions; integrate with student profile.
FR-SUP-04	L	Notifications shall alert counselors/parents when wellbeing cases are updated.
FR-SUP-05	L	Reports summarizing wellbeing/behavior trends shall be available to principals.

3.14 File Upload and Document Management

Requirement ID	Pri	Description
FR-UP-01	M	Upload routes shall accept attachments (assignments, notices, resources) with MIME validation (images, PDFs, CSV).
FR-UP-02	M	Uploaded files shall be linked to their parent records (assignment, notice, user) via stored URLs.
FR-UP-03	L	System shall support versioning or replacement of uploaded assets with audit history.

3.15 System Administration & Audit

Requirement ID	Pri	Description
FR-ADM-01	M	Audit log endpoints shall capture CRUD actions on critical modules (users, fees, exams) with metadata (actor, timestamp, IP).
FR-ADM-02	M	Admin dashboards shall expose system health info (uptime, service statuses) per 'frontend/features.txt'.
FR-ADM-03	L	Scheduled jobs (reports, reminders) shall be configurable per tenant with enable/disable flags.
FR-ADM-04	L	System shall provide configuration endpoints for CORS whitelist, rate limits, and integration keys (email/SMS).

4. External Interface Requirements

4.1 User Interface

- Responsive single-page application with multi-role dashboards, consistent yellow/purple/amber palette (see 'frontend/features.txt').
- Navigation via React Router; supports desktop and mobile gestures.
- Student portal includes interactive pet themes, motivational quick tips, and notification center.
- Principal dashboard surfaces KPI cards (student counts, revenue, infrastructure, system health).
- Accessibility: keyboard navigation, high-contrast text, ARIA labels on interactive elements.
- Error/loading states must be visible for all async operations, with toast notifications ('react-hot-toast').

4.2 REST API Interface

- Base URL defaults to 'http://localhost:5000'; environment variable 'BACKEND_URL' per deployment.
- All secured endpoints require 'Authorization: Bearer <JWT>' header.
- Requests/responses use JSON; file uploads use 'multipart/form-data'.
- API versioning currently implicit; future requirement to prefix with '/api/v1'.
- Swagger documentation generated via 'backend/swagger.js' for endpoint discovery.

4.3 Data Interfaces

- MongoDB collections represent schools, academic entities, users, attendance, assessments, AI insights, fees, notifications, audit logs.
- CSV/XLS templates for bulk operations; parsing handled via 'csv-parser' or 'xlsx'.
- Exported reports in PDF (jsPDF) and chart images (Chart.js/Recharts).
- Integration with Cloudinary for media storage; references stored as URLs.
- Email/SMS payloads structure defined by provider SDKs (Nodemailer, SMS gateway).

4.4 Hardware Interfaces

- None specific beyond standard hosting infrastructure; optional IoT attendance devices can integrate via API (future).

4.5 Software Interfaces

- Cloudinary API for uploads/downloads.
- Optional payment gateway (placeholder).

- External notification services (SMTP, SMS).
- Browser local storage/session storage for JWT persistence with secure flags, respecting logout requirements.

5. Non-Functional Requirements

5.1 Performance

Requirement ID	Pri	Description

- API shall respond within 500 ms for 90th percentile requests under nominal load (10,000 concurrent users per school).
- Bulk upload processing shall import at least 1,000 records/minute with streaming parser.
- Dashboard charts shall load within 2 seconds after data retrieval; use pagination/lazy-loading for large datasets.

5.2 Scalability

Requirement ID	Pri	Description

- Stateless API servers support horizontal scaling behind load balancers (per 'docs/architecture/01-system-architecture.md').
- Database indexes on 'schoolId', 'classId', 'studentId' ensure query scalability.
- System shall support onboarding of 50+ schools with isolated data.
- Future redis cache integration for hot datasets; SRS assumes cache invalidation strategy will follow read-through pattern.

5.3 Security

Requirement ID	Pri	Description

- JWT authentication with 24-hour expiry; refresh by login.
- Password hashing using bcrypt; secrets loaded from environment variables.
- RBAC enforced per route; unauthorized access returns 403.
- Input validation/sanitization on all payloads; reject unexpected fields.
- Audit logging for sensitive operations.
- Data encrypted in transit (HTTPS) and at rest (MongoDB Atlas).
- CORS configuration restricts origins; environment-specific whitelists.
- Rate limiting on login and bulk operations to mitigate abuse.

5.4 Availability & Reliability

Requirement ID	Pri	Description

- Target 99.5% uptime for production (basic deployment) and 99.9% for advanced architecture.
- Daily automated backups of MongoDB; point-in-time recovery for premium tiers.
- Deployment strategy should support zero-downtime rolling updates via containers or serverless hosting.
- Monitoring/alerting via Prometheus/Sentry/Pingdom (roadmap).
- Graceful degradation when external services (email/SMS) fail; log errors and show fallback UI messages.

5.5 Maintainability & Extensibility

Requirement ID	Pri	Description

- Codebase organized by feature modules (controllers/routes/models).
- ESLint and formatting enforced on frontend; consistent coding standards across backend.
- Swagger generation ensures API documentation stays synchronized.
- Architecture roadmap outlines migration to microservices; modules should minimize coupling to enable extraction.
- Configuration-driven features (e.g., enabling AI module per tenant) to reduce code changes.

5.6 Usability

Requirement ID	Pri	Description

- Mobile-first responsive design; 320px minimum viewport support.
- Consistent UI components, icons (Lucide), and color scheme.
- Provide contextual help tooltips and quick tips on dashboards.
- Provide localization readiness (string extraction) even if only English is currently implemented.

5.7 Compliance & Audit

Requirement ID	Pri	Description

- Maintain audit trail of CRUD actions, login attempts, fee edits.
- Retain student records per institutional policies (minimum 5 years).
- Support GDPR-like data export/delete requests (manual process until automation is implemented).
- Ensure parental consent flows where required for minors (policy stored per tenant).

5.8 Disaster Recovery & Backup

Requirement ID	Pri	Description

- Automated backups stored in separate region.
- Recovery Time Objective (RTO): < 4 hours; Recovery Point Objective (RPO): < 15 minutes for premium plan, < 24 hours for basic plan.
- Documented runbooks for restoring services, validated quarterly.

6. Data Requirements

- DR-01:: Every record referencing tenant data must include 'schoolId'; composite indexes '(schoolId, entityId)' for quick lookups.
- DR-02:: Student entity shall include demographic, academic, contact, guardian details aligning with 'student-bulk-upload-template.csv'.
- DR-03:: Attendance schema stores per-student entries with date, period/session, status, remarks.
- DR-04:: Fee structure/invoice/payment schemas capture currency, due dates, discounts, and payment references.
- DR-05:: AI module stores per-student subject mastery scores and generated learning path metadata.
- DR-06:: Audit log schema stores actor metadata, module, action type, payload snapshot, timestamp.
- DR-07:: File uploads store metadata (original name, MIME type, size, Cloudinary URL) and link to owning entity.

7. Appendices

7.1 Glossary

- Alcove:: In-app Q&A/community module.
- Learning Path:: AI-generated sequence of topics/resources personalized per student.
- Wellbeing Case:: Record for counseling/support interventions.
- Campus:: Sub-unit of a school (optional).
- Dashboard KPI:: Metric displayed on admin/principal dashboards (attendance %, revenue, etc.).

7.2 Traceability Notes

- Requirements map to source artifacts: API endpoints ('docs/api-reference.md'), frontend feature descriptions ('frontend/features.txt'), architectural constraints ('docs/architecture/*.md'), and sample implementations (e.g., 'backend/routes/nifStudentRoutes.js'). Future work should maintain a traceability matrix linking FR/NR IDs to user stories and test cases.

— End of Document —