PRODUCT ENGINEERING CASE STUDY

EMS — Education Management System

A Multi-Tenant SaaS Platform for End-to-End School Operations Management

PLATFORM TYPE
.NET Clean / Onion, 5-layer

ARCHITECTURE
Multi-tenant SaaS (B2B)

BACKEND STACK
ASP.NET Core, EF Core, SignalR

FRONTEND STACK
Angular (2 portals, SSR)

Table of Contents

01 Executive Summary

EMS (Education Management System) is a multi-tenant SaaS platform built to replace the fragmented, manual, and paper-driven processes that most schools rely on for academics, attendance, examinations, finance, transport, and parent communication. Rather than building a single-purpose tool, EMS was designed as a full operating system for K–12 institutions — covering five distinct user roles (SuperAdmin, School Admin, Tutor, Student, and Guardian) inside one coherent product.

The platform is engineered around a strict Clean / Onion Architecture on the backend and two purpose-built Angular frontends — one for day-to-day school operations and one dedicated SuperAdmin console for the platform owner to manage every tenant, subscription, and billing cycle from a single pane of glass.

16	33	180+	5
Core Modules	API Controllers	REST Endpoints	User Roles

02 The Problem

Schools — particularly multi-branch institutions — typically run on a patchwork of spreadsheets, WhatsApp groups, paper attendance registers, and disconnected billing tools. This creates real operational risk and makes it nearly impossible to scale operations across multiple branches or academic years consistently.

- **Fragmented data** — attendance, grades, and fee records live in different systems with no single source of truth.
- **No tenant isolation** — schools sharing infrastructure need guaranteed data separation, not just access control.
- **Academic-year complexity** — year-end transitions (promotions, repeats, withdrawals) are manual and error-prone.
- **Limited visibility** — school owners and platform operators lack real-time dashboards for finance, attendance, or churn.
- **Poor communication** — parents and tutors rely on informal channels instead of a structured, auditable system.

03 Solution Overview

EMS consolidates the entire school lifecycle — from enrollment to examinations to graduation — into a single, role-aware platform. The system is split cleanly along two axes: **who** is using it (SuperAdmin, Admin, Tutor, Student, Guardian) and **what** they are doing (academics, finance, communication, or administration). Every module respects the same multi-tenant and academic-year boundaries, so data never leaks across schools or years.

The product ships as a SaaS offering: the SuperAdmin portal lets the platform owner onboard new schools, assign subscription plans, gate premium features, and monitor platform-wide health, while each school operates independently within its own secured tenant boundary.

04 System Architecture

The backend follows a strict Clean Architecture (Onion) pattern across five projects, enforcing a one-directional dependency flow so that core business rules never depend on infrastructure details. This keeps the domain model framework-agnostic and makes the system far easier to test, extend, and refactor as the product grows.

Project	Responsibility
EMS.Domain	Core domain entities, repository interfaces, enums, and shared utilities — the innermost, dependency-free layer.
EMS.Core	Service interfaces, ViewModels/DTOs, and business logic orchestration.
EMS.Data	EF Core DbContext, repository implementations, and database migrations.
EMS.IOC	Dependency injection container wiring across all layers.
EMS.API	REST controllers, middleware, SignalR hubs, and request filters — the outermost layer.

Two Frontend Applications

- **School Portal** — a standalone Angular application serving public marketing pages (landing, about, blog, careers) alongside the authenticated back-office used by admins, tutors, and students.
- **SuperAdmin Portal** — a dedicated Angular application with server-side rendering (SSR), used exclusively by the platform owner to manage schools, subscription plans, and global settings.

Both frontends consume the same versioned REST API, ensuring a single backend contract serves both operational and platform-administration surfaces without duplicated business logic.

05 Multi-Tenancy Model

EMS uses a **company-based multi-tenancy model** with an additional academic-year scope layered on top — a pattern well suited to schools, where data isn't just partitioned by organization but also by year-over-year academic cycles.

- **Dual-scoped isolation** — every entity implements an IHasMultiTenant interface carrying both CompanyId and AcademicYearId.
- **Automatic enforcement** — global EF Core query filters silently scope every query to the correct tenant and year, removing the risk of accidental data leakage in application code.
- **Token-embedded context** — JWTs carry CompanyId and AcademicYearId directly, so tenant context travels with every authenticated request.
- **Cross-tenant visibility** — SuperAdmin accounts bypass tenant scoping to view and manage data across all schools on the platform.

This design means tenant isolation is enforced at the data-access layer itself rather than relying on developers to remember to filter by school in every query — a meaningfully stronger security guarantee for a SaaS product handling student and financial records.

06 Core Platform Modules

EMS is organized into sixteen functional modules spanning the full school operations lifecycle. Each module is independently scoped within the multi-tenant boundary but integrates with adjacent modules — for example, timetable slots feed both attendance and online-class scheduling.

Module	Core Capability
School & Academic Structure	Multi-branch company management, academic-year freeze/unfreeze controls, class/section/subject hierarchy, configurable periods, and room inventory.
User & Identity	ASP.NET Identity extended with school-specific fields, five-role model, JWT auth with 3-hour expiry and instant revocation via security stamps.
Student Management	Full lifecycle CRUD, bulk year-end promotion with history tracking, elective subject mapping, and academic-year data migration.
Tutor Management	Qualification, specialization, experience, and salary tracking, plus class-subject-tutor assignment for timetabling.
Timetable & Scheduling	Recurring and date-range schedules, special-day exception handling, conflict-free availability checks, and combined-class support.
Attendance	Daily class/section attendance and dedicated online-class attendance with manual and automatic tracking.
Examinations	Exam creation, subject-to-exam mapping with room assignment, invigilator/checker staff assignment, and paper-status lifecycle.
Results & Grading	Per-subject mark entry, configurable grade-boundary scales, and pass/fail/absent result status.
Homework	Assignment creation with attachments and due dates, a six-stage submission lifecycle, and tutor review/grading.
Fee & Finance	Class-based fee structures, discount approvals, automated late-fee engine, multi-method payment recording, student wallets, expense tracking, and payroll.
Transport	Vehicle and driver management with route and capacity tracking, plus student-to-vehicle assignment.
Communication	SignalR-powered real-time messaging, audience-targeted announcements, and read-tracked notifications.
Online/Virtual Classes	Meeting scheduling with class/subject mapping and automatic or manual attendance capture.
Subscription & Billing	Configurable plans with feature flags, billing-cycle management, and endpoint-level feature gating.
Automation & Integration	Two-way n8n webhook automation, WhatsApp Cloud API notifications, and an AI chat assistant endpoint.
SuperAdmin Platform	Platform-wide analytics dashboard, school provisioning, subscription plan administration, and activity-log auditing.

07 API Surface & Integrations

The backend exposes a single, consistent REST contract of 33 controllers and roughly 180+ endpoints, grouped into four functional tiers:

- **Public endpoints** — login, registration, AI chat, and company lookups, accessible without authentication.
- **Authenticated CRUD endpoints** — full entity management with built-in search and pagination across all modules.
- **Internal endpoints** — API-key protected routes reserved for n8n automation callbacks.
- **Master data endpoints** — enum and lookup values that drive dropdowns consistently across both frontends.

Automation & Real-Time Layer

- **n8n webhook integration** — two-way automation — EMS fires outbound webhooks on key events, and n8n can call back into protected internal APIs to complete workflows.
- **WhatsApp Cloud API** — delivers notifications directly to guardians and staff through a channel they already use daily.
- **AI integration** — a dedicated chat endpoint powers AI-assisted interactions within the platform.
- **SignalR hubs** — drive real-time messaging using a custom UserIdProvider, authenticating WebSocket connections via JWT passed in the query string.

08 Key Technical Decisions

Several deliberate, opinionated choices shape the codebase. Rather than reaching for popular defaults, the team optimized for explicitness, predictable performance, and tight control over data integrity:

Decision	Rationale
No AutoMapper	DTO mapping is written by hand in services, trading boilerplate for full visibility into every field transformation.
No CQRS / MediatR	a direct service-invoker pattern keeps the request path simple and easy to trace for a team-sized codebase.
No generic repository	hand-written repositories per entity allow precise, query-specific LINQ rather than a one-size-fits-all abstraction.
No cascading deletes	DeleteBehavior.Restrict on all foreign keys forces explicit, intentional cascade logic in repositories — preventing accidental data loss.
Client-side GUID generation	ValueGeneratedNever() on all primary keys lets IDs be generated before insert, simplifying offline-friendly and batched operations.
Academic-year freeze	a SaveChangesAsync override blocks any write against a frozen academic year, enforcing historical-record integrity at the data layer.

09 Security & Access Control

Access control in EMS is layered: authentication establishes identity, custom authorization filters establish ownership, and the subscription system gates feature access — all enforced server-side before any business logic executes.

- **JWT authentication** — tokens expire after 3 hours and embed both CompanyId and AcademicYearId for tenant-aware authorization.
- **Instant revocation** — ASP.NET Identity security-stamp validation invalidates compromised or stale tokens immediately, without waiting for natural expiry.
- **Ownership filters** — [AdminOwnership], [SuperAdminOwnership], [TutorOwnership], and [StudentOwnership] attributes enforce that users can only act on records they're entitled to.
- **Feature gating** — the [SubscriptionFeature] attribute restricts entire endpoints based on a school's active subscription plan, enabling clean tiered pricing.
- **Five-role model** — SuperAdmin, Admin, Tutor, Student, and Guardian roles map cleanly to distinct UI surfaces and API permission sets.

10 Impact & Outcomes

By unifying academics, attendance, finance, transport, and communication into one tenant-isolated platform, EMS replaces what would otherwise be five or six disconnected tools with a single system of record. The architecture decisions — strict layering, enforced tenant isolation, and an explicit no-magic approach to mapping and deletes — translate directly into outcomes that matter for a SaaS product serving multiple schools on shared infrastructure:

- **Operational consolidation** — a single platform replaces spreadsheets, messaging apps, and manual registers across the entire school lifecycle.
- **Data integrity by design** — global query filters and restricted cascading deletes make accidental cross-tenant leakage or silent data loss structurally difficult.
- **SaaS-ready monetization** — first-class subscription plans and feature gating allow the platform owner to run a tiered pricing model from day one.
- **Audit-friendly history** — academic-year freezing and promotion-history tracking preserve a defensible historical record for compliance and reporting.

11 Roadmap & Closing Notes

EMS's modular, layered foundation leaves clear room to extend the platform without disrupting existing tenants — additional automation workflows through n8n, deeper AI-assisted features building on the existing chat endpoint, and expanded analytics on the SuperAdmin dashboard are natural next steps given the architecture already in place.

"A well-isolated multi-tenant core and a strict layering discipline are what let a school platform scale from one institution to a hundred without rewriting the foundation."