**School Information System (SIS)  
Unified Business & Design Requirements (Phases A–F)**

*Version 1.0 • Audience: Stakeholders, Product, Design, Engineering, Data, QA, Security*

# 1. Executive Summary

School Information System (SIS) - Business Requirements Document  
1. Executive Summary  
This document outlines the business requirements for the development of a next-generation School Information System (SIS) designed to serve the needs of administrators, teachers, specialists, students, and parents. The SIS aims to integrate core operational functions—such as student information, scheduling, attendance, grading, and communication—into a unified, intuitive platform. The project is divided into multiple phases (A–E) to ensure structured implementation, reduce development risk, and allow for early user feedback and incremental improvements.

# 2. Background & Current Challenges

& Current Challenges  
Current SIS solutions in use across schools, particularly private K–8 institutions, often fail to provide seamless integration between administrative, instructional, and parental engagement functions. Teachers are frequently required to use multiple disconnected tools, resulting in inefficiencies, duplicated data entry, and poor year-to-year continuity. Parents often receive fragmented and delayed information about their children’s progress, while administrators lack real-time insights into school-wide performance metrics.  
  
From both teacher and parent perspectives, current systems present significant usability issues. Teachers report that the platforms are cumbersome and not tailored to the realities of daily classroom management, while parents find the available portals unhelpful in tracking progress or addressing concerns in a timely manner. Administrators struggle with data silos, limited reporting functionality, and lack of analytics to inform staffing or curriculum decisions.

# 3. Project Objectives & Goals

& Goals  
The primary objective of this project is to deliver a best-in-class SIS platform that:  
• Enhances collaboration between teachers, parents, and administrators.  
• Centralizes key academic and operational data into one secure system.  
• Provides intuitive, role-based interfaces tailored for each user type.  
• Streamlines administrative tasks such as scheduling, grading, and reporting.  
• Offers powerful analytics for tracking student performance and teacher impact year over year.  
• Integrates specialists into the scheduling and reporting structure from the start.  
• Supports both structured (middle school) and flexible (grade school) scheduling models.  
• Lays the groundwork for AI-assisted recommendations in scheduling, workload balancing, and student support.

# 4. Scope & Phase Overview

Phases A–F are delivered sequentially, with explicit dependencies and forward linkages.  
• Phase A – Core SIS foundation: users/roles, classes, scheduling, attendance, base gradebook, specialists.  
• Phase B – Curriculum & Lesson Planning tightly integrated with schedules and assignments.  
• Phase C – Advanced Grading & Assessment with online delivery, rubrics, and analytics hooks.  
• Phase D – Communication & Engagement across in‑app, email, SMS, push; parent portal.  
• Phase E – Reporting, Analytics, Insights with governed metrics and intervention loops.  
• Phase F – Future & Integrations: SSO, OneRoster/Ed‑Fi/LTI, public APIs, mobile/offline, marketplace.

# Phase A – Core SIS Foundation

## Overview & Goals

Phase A establishes the core SIS foundation, including roles, rosters, scheduling, attendance, and a flexible gradebook, with specialist pull‑out workflows. It emphasizes low‑friction teacher UX and admin‑grade scalability.

## Functional Requirements

### Roles & Permissions

Principal (full), Vice Principal (near‑full), Dean (conduct/attendance), Staff Admin (configurable), Teacher (classes/attendance/gradebook/approvals), Specialist (slots, pull‑outs, session attendance).

### Student Information

Centralized, longitudinal records: demographics, guardians, medical alerts, accommodations.

### Scheduling

Admin‑built core; teacher adjustments; specialist requests any time; teacher approval with denial reasons and alt‑slot proposal; capacity caps; conflict detection.

### Attendance

Default Present if unmarked; teacher/staff entry; specialist session logs reconcile with homeroom.

### Gradebook Baseline

Category groupings; quick entry modes; late/missing flags; extra credit; audit trail; parent‑shareable comments.

## Non‑Functional Requirements

Security (RBAC, encryption), Performance (sub‑second common ops), Reliability, Audit logging, Accessibility (WCAG 2.1 AA), Scalability to district level.

## Roles & Permissions

Role matrix per Add‑On Package; Staff Admin inherits Principal with restricted sensitive fields; granular scopes planned for future IAM.

## Core Workflows

Start‑of‑year setup; Teacher daily attendance and grading; Specialist pull‑out request and approval with alternatives; Parent portal view of grades/attendance.

## Dependencies

Auth/SSO; initial school config. Feeds Phase B (lessons), C (assessments), D (digests), E (facts/metrics), F (APIs/interop).

## Technical Constraints & Data Considerations

Stable UUIDs; core tables for users, classes, enrollments, calendar\_blocks, attendance, gradebook\_categories, assignments, grades; indexes on hot paths; audit tables; soft‑delete where appropriate.

## UI/UX Guidance

School‑themed, minimal UI; calendar day/week/month; gradebook grid with category chips; attendance quick mark; student profile tabs.

## Acceptance Criteria & KPIs

Attendance completion p50 ≤2m; grade entry 25 students ≤5m; conflicts resolved ≥99%; weekly parent engagement ≥70%.

## Appendix

### Workflow Diagrams (text)

Admin → create school → import roster → define schedules; Specialist → set slots/capacity → request; Teacher → approve/deny with reason or propose alternative.

### Example Data Structures

classes(id, teacher\_id, school\_id,...), calendar\_blocks(id, class\_id, start, end), gradebook\_categories(id, class\_id, name, weight), assignments(...), grades(...).

### Error Handling

On scheduling conflicts, surface blocking entity and suggest next open slots. On grade save failures, queue and retry; never lose inline edits.

### Edge Cases

Mid‑year transfers merge records; dual‑teacher sections; concurrent specialist sessions within capacity cap.

# Phase B – Curriculum & Lesson Planning

## Overview & Goals

Phase B delivers curriculum maps, unit and lesson planning tightly integrated with schedules and gradebook. Teachers plan once; data flows to assignments and parent digests.

## Functional Requirements

### Planning Levels

Curriculum maps (year/term) → Unit plans (multi‑week) → Lesson plans (daily/block) with standards alignment.

### Templates & Reuse

Built‑in + school custom templates; copy/clone across dates/years; version history; draft/publish; lock semantics.

### Standards Library

Searchable sets; coverage tracking vs taught; alignment feeding Phase C mastery.

### Schedule Integration

Drag lessons to calendar blocks; shifts follow block moves; attachments on calendar; take attendance from block.

### Gradebook & Assessment Links

Create assignments from lessons; link to assessments; flow standards to analytics.

### Collaboration & Sharing

Co‑teaching, team libraries, comments/@mentions, suggestion mode; parent‑friendly summaries.

## Non‑Functional Requirements

Autosave <500ms; open lesson <1.5s p95; copy 20 lessons <5s; version history ≥3y; role-based visibility; encrypted internal fields.

## Roles & Permissions

Teachers author; team leads review; admin read‑only oversight; parents see summaries only; specialists annotate interventions aligned to lessons.

## Core Workflows

Year/Term setup → teachers receive maps → author units/lessons → link to calendar → create assignments → publish parent summaries.

## Dependencies

Requires Phase A classes/schedules/gradebook; feeds Phase C assessments and Phase D digests; supports Phase E longitudinal analysis.

## Technical Constraints & Data Considerations

Entities: curriculum\_maps, unit\_plans, lesson\_plans, lesson\_schedule\_links, lesson\_assignment\_links, templates, standards; integrity checks to prevent orphans; versioning metadata.

## UI/UX Guidance

Planner with week timeline; lesson editor with objective/activities/minutes panels; standards picker; parent summary preview; global search and tagging.

## Acceptance Criteria & KPIs

Median authoring time ≤10m; reuse rate ≥40%; standards coverage variance ≤10%; parent clarity ≥80%.

## Appendix

### Workflow Diagrams (text)

Teacher opens planner → selects template → drafts objectives → aligns standards → drags onto calendar → creates assignment from lesson → publishes summary.

### Example Data Structures

lesson\_plans(id, class\_id, date, duration, objectives, activities[], standards[], attachments[], status, version).

### Error Handling

Broken standard refs surface remediation; schedule moves prompt shift options; merge conflicts show diff with accept/reject.

### Edge Cases

Split classes; day rotations; missing standards mapped later with audit trail.

# Phase C – Advanced Grading & Assessment

## Overview & Goals

Phase C upgrades grading into a comprehensive assessment platform: online delivery, rubrics, auto‑grading, retakes, and deep analytics hooks—all aligned to standards and schedules.

## Functional Requirements

### Gradebook Management

Define categories/weights; flexible scales (points, % , letter, standards‑based); bulk/category-first entry; late/missing policies; grade overrides with reasons.

### Assessment Delivery

Student portal for submissions; supported item types: MCQ, TF, SA, FITB, rubric-based; auto‑grading with partial credit; manual override; anti‑cheat options (randomization, time windows).

### Rubrics & Standards

Reusable rubrics; criterion scoring; standards tagging per item; mastery roll‑ups.

### Reassessments & Retakes

Highest/latest/average policies per assessment; configurable cooldown windows; proctor flags; full history.

### Cross‑Class Operations

Bulk create/assign across sections; blueprint reuse year‑to‑year with versioning.

### Parent/Student Views

Timely feedback, comments, missing/late flags; resubmission windows; study resources linked to missed standards.

## Non‑Functional Requirements

Auto‑grade latency ≤2s/class; concurrency for 5k simultaneous submissions; WCAG 2.1 AA; FERPA; detailed audit trails for score changes.

## Roles & Permissions

Teacher (full for own classes), Admin (policies & audits), Specialist (limited to caseload), Parent/Student (read only).

## Core Workflows

Create assessment → align to standards → schedule (avoids conflicts using Phase A calendar) → deliver online → auto‑grade → review/override → publish → notify via Phase D.

## Dependencies

Requires Phase A (rosters, schedules), Phase B (lessons/standards). Feeds Phase D alerts and Phase E item/mastery analytics, Phase F grade return APIs.

## Technical Constraints & Data Considerations

Entities: assessments, items, attempts, responses, scoresheets, rubrics; calculation rules for weighting/retakes; item analysis storage; indexes on attempts(student\_id, assessment\_id).

## UI/UX Guidance

Teacher authoring UI with item banks; student secure test mode; grading queue with filters; rubric matrix view; progress and integrity indicators.

## Acceptance Criteria & KPIs

90%+ weekly gradebook usage; ≥50% assessments online Y2; parent login ≥75%.

## Appendix

### Workflow Diagrams (text)

Teacher drafts assessment → adds items/rubric → sets window → students take → system auto‑grades → teacher reviews exceptions → publish grades.

### Example Data Structures

assessments(id, class\_id, window\_start/end, retake\_policy,...); items(id, type, standard\_code,...); attempts(id, student\_id, started\_at, submitted\_at, score).

### Error Handling

Network loss queues submissions; conflicting retake rules blocked; invalid item configs flagged pre‑publish.

### Edge Cases

Transfers mid‑term; accommodations (extra time, text‑to‑speech); offline packets with later sync.

# Phase D – Communication & Engagement

## Overview & Goals

Phase D consolidates all communications—messages, announcements, forums, alerts—and powers parent engagement with multilingual, accessible, and auditable channels.

## Functional Requirements

### Messaging & Announcements

Role‑aware threads; class/school broadcasts; attachments; read receipts; translation; templates; merge fields.

### Discussion Boards

Class and school forums; moderation; tagging and search; student participation metrics; announcement‑only mode.

### Alerts & Emergencies

Priority tiers; multi‑channel fan‑out (push/email/SMS); geo‑fencing; admin override of preferences for emergencies.

### Parent Portal

Unified child dashboard; calendar of assignments/events; surveys/polls; digest preferences.

### Teacher/Admin Tools

Quick actions (e.g., message parents of absentees); scheduled newsletters; engagement analytics; RBAC and audit logs.

## Non‑Functional Requirements

Delivery p95 < 5s for push; SMS/email provider retries; translation quality with human override; WCAG 2.1 AA; FERPA consent preferences.

## Roles & Permissions

Admin (broadcast & policy), Teacher (class messaging), Specialist (caseload), Parent (child threads), Student (teacher/specialist only).

## Core Workflows

Event triggers (absence, missing work, low grade) → compose from template → send via channels → track deliveries/opens → nudge or escalate.

## Dependencies

Depends on A (roster), B (lesson summaries), C (grades); feeds E engagement metrics; integrates with F notification connectors.

## Technical Constraints & Data Considerations

Entities: threads, messages, deliveries, digests, preferences; webhooks for delivery receipts; rate limiting and anti‑spam; retention policies.

## UI/UX Guidance

Unified inbox with filters; notification bell with badges; compose with translation preview; parent digest layout.

## Acceptance Criteria & KPIs

Parent portal MAU +20%; median reply latency ↓; delivery success ≥ 99%.

## Appendix

### Workflow Diagrams (text)

Teacher selects 'absent today' cohort → sends note to parents → system logs opens → unresolved get follow‑up after 24h.

### Example Data Structures

messages(id, thread\_id, sender\_id, recipients[], channel, body, lang, sent\_at); deliveries(id, status, provider\_ref).

### Error Handling

Bounced emails retried with backoff; invalid numbers quarantined; opt‑out honored per policy.

### Edge Cases

Split guardianship with different languages and preferences; emergency overrides logged.

# Phase E – Reporting, Analytics & Insights

## Overview & Goals

Phase E delivers governed analytics: dashboards for every persona, explainable alerts, and intervention loops—without busywork.

## Functional Requirements

### Dashboards

District/school/teacher/student 360 with drill‑downs and equity slices; time contexts and trends.

### Teacher Analytics

Mastery heatmaps; workload and time‑to‑grade; actionable priority lists.

### Student 360 & Interventions

Unified views with plans, dosage, owners, outcomes; impact scoring.

### Standards & Curriculum Analytics

Coverage vs mastery; pacing adherence; resource effectiveness.

### Assessment & Gradebook Analytics

Item analysis; weight/extra‑credit sensitivity; retake policy outcomes.

### Attendance & Engagement

Chronic absence, correlations, portal and messaging signals.

### Alerts & Subscriptions

Threshold rules with explainability; smart digests; deep‑links to actions.

### Exports & APIs

Self‑service CSV/Excel/PDF; secure aggregates via REST/GraphQL; scheduled delivery.

## Non‑Functional Requirements

Initial dashboard ≤3s p95; heavy reports ≤7s; exports ≤60s; near‑real‑time freshness class‑level; hourly school‑level; daily district; RLS/CLS security; WCAG 2.1 AA.

## Roles & Permissions

Leaders (district/school), Teachers, Specialists, Parents/Students (simplified).

## Core Workflows

Alert fires → create intervention → follow‑ups scheduled via D → outcomes measured → analytics update → close/adjust plan.

## Dependencies

Consumes A–D; exposes to F via contracts. Requires governed metrics catalog and lineage.

## Technical Constraints & Data Considerations

Warehouse facts: grades, assessments, attendance, schedule, pull‑outs, communications, interventions; conformed dimensions; lineage and quality checks; semantic layer metrics dictionary.

## UI/UX Guidance

Configurable dashboards; explainability tooltips; slice-and-dice filters; accessible tables and charts; subscriptions UI.

## Acceptance Criteria & KPIs

Weekly admin reporting time −30%; intervention start time −25%; chronic absence −15%; parent digest usage +20%.

## Appendix

### Workflow Diagrams (text)

Data events stream → curated facts → semantic layer → dashboards/alerts → intervention flows.

### Example Data Structures

fact\_grades(...), fact\_assessments(...), dim\_student(SCD2...), dim\_calendar(...).

### Error Handling

Late data watermarking; backfills tagged; metric formula versioning with approvals.

### Edge Cases

Roster changes mid‑term; subgroup reclassification; partial data outages → degraded summaries.

# Phase F – Future & Integrations

## Overview & Goals

Phase F operationalizes interoperability and scale: standards‑based rostering, SSO, LTI, public APIs, events/webhooks, mobile/offline, and a safe extension model.

## Functional Requirements

### Identity & Access

SSO (SAML/OIDC), SCIM provisioning, role and class claims; consent preferences.

### Rostering & SIS Interop

OneRoster 1.1+/REST & CSV, Ed‑Fi v5.x core; Clever/ClassLink sync and launches.

### LMS & Classroom Tools

LTI 1.3/Advantage deep links; Assignment & Grade Service back to LMS; Google/Microsoft summaries.

### State/District Reporting

Schema‑mapped exports/APIs; validations; secure SFTP/API delivery.

### Eventing/Webhooks

Domain events (assignment.graded, attendance.marked); retries, signatures, replay window.

### Import/Export Pipelines

Guided mappers, preview/validate, idempotent upserts, diff‑aware updates, error queues.

### Plugin/Extension Model

Embeddable panels with scoped tokens; audit trails; marketplace governance.

### Mobile & Offline

Native apps with push; offline queues for attendance/grades; kiosk mode roadmap.

### AI Assistants (Guardrailed)

Scheduling suggestions, student 360 summaries, authoring helpers with explainability and human‑in‑the‑loop.

## Non‑Functional Requirements

API p95 <300ms; webhook delivery ≥99% within 60s; import ≥50k rows/min; SOC2 roadmap; PII minimization/DLP; observability, rate limits, DR (RPO ≤15m, RTO ≤2h).

## Roles & Permissions

Tenant admins manage connections; developers use APIs; all app roles benefit from SSO and data flows.

## Core Workflows

Onboard tenant → connect IdP → run OneRoster import dry‑run → fix mappings → commit → verify dashboards → enable LTI → pilot grade return.

## Dependencies

Assumes A–E contracts are stable; publishes contracts and SDKs; ensures backward compatibility with versioning and long deprecation windows.

## Technical Constraints & Data Considerations

Canonical contracts for District, School, Term, Calendar, User, Role, Class, Enrollment, Assignment, Grade, Attendance, Assessment, Attempt, Standard, Lesson, Resource, Intervention, Message, Event; crosswalks; view tokens.

## UI/UX Guidance

Integration Center with connectors, job logs, diagnostics, and mapping UI; webhook monitor; API key management; extension gallery.

## Acceptance Criteria & KPIs

Roster maintenance −50%; ≥90% tools connected via standards; integration failure ≤1%/mo (auto‑recovered).

## Appendix

### Workflow Diagrams (text)

Admin opens Integration Center → selects connector → maps fields → validates sample → schedules sync → monitors jobs/webhooks.

### Example Data Structures

webhooks(id, topic, target\_url, hmac\_secret, delivery\_status), external\_mappings(external\_id, internal\_id, transform).

### Error Handling

Dead‑letter queues; replay; schema drift detection; rate‑limit backoff; secrets rotation reminders.

### Edge Cases

Dual‑source rosters; partial state extracts; timezone skew in timestamps; consent revocation impacts exports.