School Information System (SIS) — Design & Architecture (Next.js)

> Versi: 1.2 (gabungan semua modul, lengkap dengan skema database) Target: Web-based app berbasis Next.js. Modul: **Absensi Siswa & Pegawai**, **Raport**, **Data Siswa & Kurikulum**, **PPDB**, **Keuangan**, **Tabungan Siswa**, **Perpustakaan**, **Aset**, **Ekstrakurikuler**, **BK/Konseling**, **CBT**, **Analitik**, **Notifikasi WA**.

---

1) Tujuan & Ruang Lingkup

Tujuan: Membangun SIS modular yang komprehensif, siap dipakai semua peran di sekolah, transparan untuk orang tua/siswa, dan mudah dikembangkan. Disusun agar langsung dapat dieksekusi oleh GPT Codex/VSCode.

Ruang lingkup: Semua modul digabung: Master Data, Absensi (siswa & pegawai), Penilaian & Raport, PPDB, Keuangan & Tabungan, Perpustakaan, Aset, Ekstrakurikuler, BK, CBT, Analitik, Notifikasi WA/ Email, Portal publik.

---

2) Persona & RBAC

Peran inti: - Super Admin. - Kepala Sekolah. - Admin TU/Operator. - Bendahara (keuangan & tabungan). - HR/Kepegawaian (pegawai, shift, absensi, cuti, lembur). - Guru. - Wali Kelas. - Staf (non-guru). - Siswa. - Orang Tua/Wali. - Pustakawan. - Petugas Aset. - Pembina Ekstrakurikuler. - Guru BK.

Akses inti: - Absensi Siswa: guru/wali input, wali/kepsek approve. - Absensi Pegawai: HR atur shift, pegawai check-in/out, HR approve koreksi. - Keuangan & Tabungan: Bendahara. - Perpustakaan: pustakawan. - Aset: petugas aset. - Ekstra: pembina. - BK: guru BK. - CBT: admin/guru. - Notifikasi: admin/super.

---

3) Arsitektur Aplikasi

- **Frontend/Fullstack**: Next.js (App Router) + API Routes.
- **UI/State**: Tailwind + shadcn/ui, React Query, Zustand.
- **Auth**: NextAuth (JWT) + RBAC middleware.
- **Database**: PostgreSQL/MySQL via Prisma.
- **Storage**: MinIO/S3.
- **Export**: React-PDF/Puppeteer.
- **Infra**: Docker Compose dev, VPS prod (Caddy/NGINX).
- **CI/CD**: GitHub Actions.
- **Observability**: logging, metrics, uptime ping.

---

4) Modul & Fitur (Lengkap)

1. Master Data.
2. Absensi (Siswa & Pegawai).
3. Penilaian & Raport.
4. PPDB.
5. Keuangan (SPP, Tagihan, laporan, diskon, beasiswa, refund).
6. Tabungan Siswa.
7. Perpustakaan (koleksi, pinjam/kembali, denda, barcode/RFID).
8. Aset (inventaris, pinjam, maintenance).
9. Ekstrakurikuler (pendaftaran, presensi, nilai, event).
10. BK/Konseling (kasus, sesi, catatan, rujukan).
11. CBT/Ujian Online (sinkron LMS, soal, hasil ujian).
12. Analitik & Dashboard.
13. Notifikasi WA/Email.
14. Portal Publik.

---

5) Peta Halaman

- Publik: Home, Berita, Agenda, Galeri, Profil, Kontak, PPDB.
- Auth: Login, Register, Forgot.
- Dashboard: Master, Absensi (Siswa & Pegawai), Penilaian, Raport, PPDB, Keuangan, Tabungan, Perpustakaan, Aset, Ekstra, BK, CBT, Analitik, HR.
- Portal Siswa/Ortu: jadwal, nilai, raport, presensi, tagihan, tabungan, notifikasi.
- Portal Pegawai: check-in/out, timesheet, cuti/izin.

---

6) Model Data & ERD (detail database)

Akademik

- `schools`
- `academic_years`
- `semesters`
- `grades`
- `classes`
- `subjects`
- `curricula`
- `curriculum_subjects`
- `teachers`
- `students`
- `enrollments`
- `schedules`
- `attendance` (siswa)
- `assessments`
- `report_cards`

Kepegawaian

- `employees`
- `shifts`
- `employee_shifts`
- `attendance_staff`
- `leave_types`
- `leave_requests`
- `overtimes`

PPDB

- `admissions`
- `admission_documents`
- `admission_scores`

Keuangan

- `fee_rules`
- `invoices`
- `invoice_items`
- `payments`
- `payment_attempts`
- `discounts`
- `scholarships`
- `refunds`
- `cashbooks`

Tabungan

- `savings_accounts`
- `savings_transactions`
- (view) `savings_balances`

Perpustakaan

- `lib_items`
- `lib_members`
- `lib_loans`
- `lib_barcodes`
- `lib_settings`

Aset

- `assets`
- `asset_loans`
- `asset_maintenance`

Ekstrakurikuler

- `extracurriculars`
- `extracurricular_members`

- `extracurricular_attendance`
- `extracurricular_events`

BK/Konseling

- `counseling_tickets`
- `counseling_sessions`
- `counseling_refs`

CBT/Ujian

- `lms_links`
- `lms_scores`
- `exams`
- `exam_questions`
- `exam_attempts`

Analitik & Notifikasi

- `events`
- `wa_templates`
- `wa_outbox`

CMS

- `posts`
- `events`
- `media`

Auth & RBAC

- `users`
- `roles`
- `permissions`
- `role_permissions`

---

7) Alur Kunci

- Absensi Siswa.
- Absensi Pegawai.
- Raport.
- PPDB.
- Keuangan.
- Tabungan.
- Perpustakaan.
- Aset.
- Ekstrakurikuler.
- BK.
- CBT.
- Notifikasi WA.

• Analitik.

---

8) Kontrak API (High-level)

• Auth: /auth.
• Master: /academic-years, /semesters, /grades, /classes, /subjects, /curricula, /teachers, /students.
• Absensi: /attendance (siswa), /hr/* (pegawai).
• Penilaian & Raport: /assessments, /report-cards.
• PPDB: /admissions.
• Keuangan: /finance/*.
• Tabungan: /savings/*.
• Perpustakaan: /library/*.
• Aset: /assets/*.
• Ekstrakurikuler: /extras/*.
• BK: /counseling/*.
• CBT: /lms/*, /exams/*.
• Notifikasi: /wa/*.
• Analitik: /analytics/*.

---

9) UX & Design System

• Komponen: AppShell, Navbar, Sidebar, DataTable, Form, Modal, Stepper, Charts, Toast.
• Halaman ringkas per modul.
• Aksesibilitas AA, mobile-first, i18n.

---

10) Rencana Eksekusi (Tanpa Fase)

> Semua modul **masuk scope**. Eksekusi **paralel** berdasarkan **prioritas teknis & dependensi**, bukan fase.

Prioritas 0 – Fondasi - Skeleton Next.js + Auth (NextAuth) + RBAC middleware. - Prisma schema inti + migrasi + seeder. - Komponen UI dasar (AppShell, DataTable, Form, Toast) + util (Zod, logger).

Prioritas 1 – Inti Akademik & HR - Master Data (akademik & user/role) → Absensi **Siswa & Pegawai** (shift, check-in/out, koreksi, timesheet). - Penilaian dasar (komponen nilai) & generator Raport (PDF, publish).

Prioritas 2 – Administratif - PPDB end-to-end (apply → verify → decide → enroll) + impor data. - Perpustakaan (pinjam/kembali, denda) + Aset (register, peminjaman, maintenance). - Ekstrakurikuler (pendaftaran, presensi, event) + BK/Konseling (tiket, sesi, catatan).

Prioritas 3 – Finansial & Tabungan - Keuangan: fee rules, generate invoice massal, pembayaran (cash/gateway), laporan aging. - Tabungan Siswa: akun, setoran/penarikan (approval), buku tabungan PDF.

Prioritas 4 – Integrasi & Observabilitas - Integrasi CBT/LMS (sinkron, impor nilai). - Notifikasi WA/ Email (template, outbox, webhook, retry). - Analitik & Dashboard (KPI presensi, nilai, keuangan, tabungan, PPDB).

Prioritas 5 – Hardening - Rate limit, audit log, backup/restore, e2e testing Playwright, CI/CD lengkap.

> Catatan: Item dapat dikerjakan **overlap** selama dependensi terpenuhi.

11) Pedoman Implementasi (untuk Codex/VSCode)

- Stack: Next.js, Prisma, NextAuth, Zod, RHF, React Query, Tailwind, shadcn/ui.
- Struktur:

```
src/
  app/(public|auth|dashboard)/*
  app/api/*
  components/*
  lib/{auth,db,rbac,validators,wa,analytics}.ts
  features/
{attendance,hr,assessments,report,ppdb,finance,savings,library,assets,extras,counseling,
*
prisma/schema.prisma
prisma/seed.ts
```

- Konvensi: kebab-case file, camelCase var/fungsi, PascalCase komponen.
- Commit: Conventional Commits.
- Validasi: Zod.
- RBAC: middleware guard.
- Transaksi: prisma.$transaction untuk keuangan/tabungan.
- WA: sendWA(templateKey, to, vars) → enqueue wa_outbox → worker kirim.
- Testing: unit, integration, e2e (Playwright).
- Dokumentasi: generate OpenAPI dari Zod.
- TODO markers untuk milestone.

12) Skema Database (Prisma) — Lengkap Siap Eksekusi

> Gunakan **PostgreSQL** (disarankan). Bisa dialihkan ke MySQL dengan penyesuaian kecil. Simpan sebagai `prisma/schema.prisma` lalu jalankan `npx prisma generate` & `npx prisma migrate dev`.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql" // ganti ke "mysql" jika perlu
  url      = env("DATABASE_URL")
}
```

```
// ===== ENUMS =====
enum StudentAttendanceStatus { PRESENT SICK PERMIT ABSENT }
enum StaffAttendanceStatus   { OK LATE EARLY ABSENT }
enum AdmissionStatus         { APPLIED VERIFIED ACCEPTED REJECTED ENROLLED }
enum InvoiceStatus           { UNPAID PARTIAL PAID VOID }
enum PaymentMethod           { CASH TRANSFER GATEWAY }
enum SavingsTxnType          { DEPOSIT WITHDRAW ADJUSTMENT }
enum WaStatus                { PENDING SENT FAILED }

// ===== AUTH & RBAC =====
model User {
  id        String   @id @default(cuid())
  name      String
  email     String   @unique
  password  String
  roleId    String?
  role      Role?    @relation(fields: [roleId], references: [id])
  teacher   Teacher?
  student   Student?
  employee  Employee?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}

model Role {
  id        String           @id @default(cuid())
  name      String           @unique
  users     User[]
  perms     RolePermission[]
}

model Permission {
  id    String           @id @default(cuid())
  name  String           @unique
  roles RolePermission[]
}

model RolePermission {
  roleId       String
  permissionId String
  role         Role       @relation(fields: [roleId], references: [id])
  permission   Permission @relation(fields: [permissionId], references: [id])
  @@id([roleId, permissionId])
}

// ===== AKADEMIK =====
model School {
  id        String @id @default(cuid())
  name      String
  address   String?
  classes   Classroom[]
```

```
}

model AcademicYear {
  id      String   @id @default(cuid())
  name    String   @unique // e.g. 2025/2026
  active  Boolean  @default(true)
  enrolls Enrollment[]
}

model Semester {
  id        String   @id @default(cuid())
  yearId    String
  term      String   // ganjil/genap
  active    Boolean  @default(true)
  year      AcademicYear @relation(fields: [yearId], references: [id])
  reports   ReportCard[]
}

model Grade {
  id      String @id @default(cuid())
  name    String
  classes Classroom[]
}

model Classroom {
  id         String  @id @default(cuid())
  schoolId   String?
  gradeId    String
  name       String
  homeroomId String?
  school     School?  @relation(fields: [schoolId], references: [id])
  grade      Grade    @relation(fields: [gradeId], references: [id])
  homeroom   Teacher? @relation("HomeroomTeacher", fields: [homeroomId],
references: [id])
  schedules  Schedule[]
  enrolls    Enrollment[]
  attendance Attendance[]
}

model Subject {
  id       String @id @default(cuid())
  code     String @unique
  name     String
  curSubs  CurriculumSubject[]
  schedules Schedule[]
}

model Curriculum {
  id      String @id @default(cuid())
  name    String @unique
  desc    String?
```

```
  subjects CurriculumSubject[]
}

model CurriculumSubject {
  id           String     @id @default(cuid())
  curriculumId String
  subjectId    String
  weightSchema Json?
  curriculum   Curriculum @relation(fields: [curriculumId], references: [id])
  subject      Subject    @relation(fields: [subjectId], references: [id])
  @@unique([curriculumId, subjectId])
}

model Teacher {
  id        String  @id @default(cuid())
  userId    String  @unique
  profile   Json?
  user      User    @relation(fields: [userId], references: [id])
  schedules Schedule[]
}

model Student {
  id        String  @id @default(cuid())
  nisn      String? @unique
  nis       String? @unique
  userId    String? @unique
  profile   Json?
  status    String? // active/graduated/transfer
  user      User?   @relation(fields: [userId], references: [id])
  enrolls   Enrollment[]
  attendance Attendance[]
  assessments Assessment[]
  reports    ReportCard[]
  invoices   Invoice[]
  savings    SavingsAccount[]
}

model Enrollment {
  id             String @id @default(cuid())
  studentId      String
  classId        String
  academicYearId String
  student        Student      @relation(fields: [studentId], references:
[id])
  class          Classroom    @relation(fields: [classId], references: [id])
  year           AcademicYear @relation(fields: [academicYearId], references:
[id])
  @@unique([studentId, classId, academicYearId])
}

model Schedule {
```

```
  id        String  @id @default(cuid())
  classId   String
  subjectId String
  teacherId String
  day       Int     // 0..6
  startTime String  // "07:00"
  endTime   String  // "08:40"
  class     Classroom @relation(fields: [classId], references: [id])
  subject   Subject   @relation(fields: [subjectId], references: [id])
  teacher   Teacher   @relation(fields: [teacherId], references: [id])
}

model Attendance {
  id        String  @id @default(cuid())
  date      DateTime
  classId   String
  studentId String
  status    StudentAttendanceStatus
  note      String?
  takenById String?
  class     Classroom @relation(fields: [classId], references: [id])
  student   Student   @relation(fields: [studentId], references: [id])
  @@index([date, classId])
}

model Assessment {
  id         String  @id @default(cuid())
  classId    String
  subjectId  String
  studentId  String
  component  String  // tugas, UH, PTS, PAS
  score      Float
  weight     Float?
  teacherId  String?
  student    Student   @relation(fields: [studentId], references: [id])
  @@index([classId, subjectId, studentId])
}

model ReportCard {
  id          String   @id @default(cuid())
  studentId   String
  classId     String
  semesterId  String
  summary     String?
  rank        Int?
  remarks     String?
  approvedBy  String?
  publishedAt DateTime?
  student     Student  @relation(fields: [studentId], references: [id])
  semester    Semester @relation(fields: [semesterId], references: [id])
  @@unique([studentId, semesterId])
```

```
}

// ===== HR / KEPEGAWAIAN =====
model Employee {
  id         String   @id @default(cuid())
  userId     String   @unique
  type       String   // guru/staf
  position   String?
  department String?
  hireDate   DateTime?
  status     String?  // active/inactive
  user       User     @relation(fields: [userId], references: [id])
  shifts     EmployeeShift[]
  staffAtt   AttendanceStaff[]
}

model Shift {
  id       String @id @default(cuid())
  name     String @unique
  startAt  String
  endAt    String
  graceIn  Int    @default(0)
  graceOut Int    @default(0)
  assigned EmployeeShift[]
}

model EmployeeShift {
  id         String  @id @default(cuid())
  employeeId String
  shiftId    String
  effectiveFrom DateTime
  effectiveTo   DateTime?
  employee   Employee @relation(fields: [employeeId], references: [id])
  shift      Shift    @relation(fields: [shiftId], references: [id])
  @@index([employeeId, effectiveFrom])
}

model AttendanceStaff {
  id          String  @id @default(cuid())
  employeeId  String
  date        DateTime
  checkInAt   DateTime?
  checkOutAt  DateTime?
  method      String?  // qr/pin/gps
  location    String?  // lat,lng
  note        String?
  status      StaffAttendanceStatus?
  employee    Employee @relation(fields: [employeeId], references: [id])
  @@index([date, employeeId])
}
```

```
model LeaveType {
  id   String @id @default(cuid())
  code String @unique
  name String
  paid Boolean @default(false)
}

model LeaveRequest {
  id           String  @id @default(cuid())
  employeeId   String
  leaveTypeId  String
  startDate    DateTime
  endDate      DateTime
  reason       String?
  status       String   @default("pending") // pending/approved/rejected
  approvedBy   String?
  employee     Employee @relation(fields: [employeeId], references: [id])
  leaveType    LeaveType @relation(fields: [leaveTypeId], references: [id])
}

model Overtime {
  id         String  @id @default(cuid())
  employeeId String
  date       DateTime
  startAt    DateTime
  endAt      DateTime
  approvedBy String?
  note       String?
  employee   Employee @relation(fields: [employeeId], references: [id])
}

// ===== PPDB =====
model Admission {
  id          String   @id @default(cuid())
  regNo       String   @unique
  name        String
  contacts    Json?
  chosenGrade String?
  status      AdmissionStatus @default(APPLIED)
  documents   AdmissionDocument[]
  scores      AdmissionScore[]
}

model AdmissionDocument {
  id          String @id @default(cuid())
  admissionId String
  docType     String
  url         String
  verifiedBy  String?
  verifiedAt  DateTime?
  admission   Admission @relation(fields: [admissionId], references: [id])
```

```
}

model AdmissionScore {
  id          String @id @default(cuid())
  admissionId String
  criteria    String
  score       Float
  admission   Admission @relation(fields: [admissionId], references: [id])
}

// ===== KEUANGAN =====
model FeeRule {
  id             String   @id @default(cuid())
  academicYearId String
  gradeId        String?
  classId        String?
  name           String
  amount         Int      // in smallest currency unit
  period         String   // bulanan/tahunan/sekali
  dueDay         Int?
  active         Boolean  @default(true)
}

model Invoice {
  id        String   @id @default(cuid())
  studentId String
  dueDate   DateTime
  status    InvoiceStatus @default(UNPAID)
  total     Int      @default(0)
  notes     String?
  items     InvoiceItem[]
  payments  Payment[]
  student   Student @relation(fields: [studentId], references: [id])
  @@index([studentId, status])
}

model InvoiceItem {
  id        String  @id @default(cuid())
  invoiceId String
  feeRuleId String?
  description String
  qty       Int     @default(1)
  unitPrice Int     @default(0)
  amount    Int     @default(0)
  invoice   Invoice @relation(fields: [invoiceId], references: [id])
}

model Payment {
  id         String   @id @default(cuid())
  invoiceId  String
  method     PaymentMethod
```

```
  paidAt     DateTime @default(now())
  amount     Int
  reference  String?
  receivedBy String?
  invoice    Invoice  @relation(fields: [invoiceId], references: [id])
}

model PaymentAttempt {
  id        String   @id @default(cuid())
  invoiceId String
  gateway   String
  status    String
  payload   Json?
  updatedAt DateTime @updatedAt
  invoice   Invoice  @relation(fields: [invoiceId], references: [id])
}

model Discount {
  id        String  @id @default(cuid())
  studentId String?
  gradeId   String?
  name      String
  type      String  // percent/flat
  value     Int
  validFrom DateTime?
  validTo   DateTime?
}

model Scholarship {
  id         String  @id @default(cuid())
  studentId  String
  name       String
  coverPct   Int?    // 0-100
  cap        Int?
  student    Student @relation(fields: [studentId], references: [id])
}

model Refund {
  id         String  @id @default(cuid())
  paymentId  String
  amount     Int
  reason     String?
  processedBy String?
  payment    Payment @relation(fields: [paymentId], references: [id])
}

model Cashbook {
  id        String   @id @default(cuid())
  date      DateTime @default(now())
  kind      String   // in/out
  amount    Int
```

```
  category  String
  memo      String?
  createdBy String?
  @@index([date, category])
}

// ===== TABUNGAN SISWA =====
model SavingsAccount {
  id         String   @id @default(cuid())
  studentId  String
  openDate   DateTime @default(now())
  status     String   @default("active") // active/closed
  minBalance Int      @default(0)
  notes      String?
  student    Student  @relation(fields: [studentId], references: [id])
  txns       SavingsTransaction[]
}

model SavingsTransaction {
  id         String   @id @default(cuid())
  accountId  String
  type       SavingsTxnType
  amount     Int
  trxDate    DateTime @default(now())
  method     String?  // cash/transfer
  reference  String?
  createdBy  String?
  approvedBy String?
  account    SavingsAccount @relation(fields: [accountId], references: [id])
  @@index([accountId, trxDate])
}

// ===== PERPUSTAKAAN =====
model LibItem {
  id               String @id @default(cuid())
  code             String @unique
  title            String
  author           String?
  publisher        String?
  year             Int?
  category         String?
  isbn             String?
  copiesTotal      Int     @default(1)
  copiesAvailable  Int     @default(1)
  barcodes         LibBarcode[]
  loans            LibLoan[]
}

model LibMember {
  id      String @id @default(cuid())
  userId  String?
```

```
  name    String
  role    String // student/teacher/extern
  user    User?  @relation(fields: [userId], references: [id])
  loans   LibLoan[]
}

model LibLoan {
  id        String   @id @default(cuid())
  itemId    String
  memberId  String
  loanDate  DateTime @default(now())
  dueDate   DateTime
  returnDate DateTime?
  fineAmount Int     @default(0)
  item      LibItem   @relation(fields: [itemId], references: [id])
  member    LibMember @relation(fields: [memberId], references: [id])
  @@index([memberId, dueDate])
}

model LibBarcode {
  id      String @id @default(cuid())
  itemId  String
  barcode String @unique
  item    LibItem @relation(fields: [itemId], references: [id])
}

model LibSetting {
  id          String @id @default(cuid())
  maxLoans    Int    @default(3)
  loanDays    Int    @default(7)
  finePerDay  Int    @default(0)
}

// ===== ASET SEKOLAH =====
model Asset {
  id          String  @id @default(cuid())
  code        String  @unique
  name        String
  category    String?
  purchaseDate DateTime?
  cost        Int?
  location    String?
  custodianId String?
  status      String? // available/loaned/maintenance
  loans       AssetLoan[]
  maints      AssetMaintenance[]
}

model AssetLoan {
  id         String  @id @default(cuid())
  assetId    String
```

```
  borrowerId String
  startDate  DateTime
  endDate    DateTime?
  returnedAt DateTime?
  asset      Asset @relation(fields: [assetId], references: [id])
}

model AssetMaintenance {
  id       String  @id @default(cuid())
  assetId  String
  date     DateTime @default(now())
  action   String
  cost     Int?
  notes    String?
  asset    Asset @relation(fields: [assetId], references: [id])
}

// ===== EKSTRAKURIKULER =====
model Extracurricular {
  id        String  @id @default(cuid())
  name      String
  coachId   String?
  schedule  String?
  members   ExtracurricularMember[]
  attends   ExtracurricularAttendance[]
  events    ExtracurricularEvent[]
}

model ExtracurricularMember {
  id                String @id @default(cuid())
  extracurricularId String
  studentId         String
  joinedAt          DateTime @default(now())
  @@unique([extracurricularId, studentId])
}

model ExtracurricularAttendance {
  id                String   @id @default(cuid())
  extracurricularId String
  date              DateTime
  studentId         String
  status            String // hadir/izin/sakit/alfa
  @@index([extracurricularId, date])
}

model ExtracurricularEvent {
  id                String   @id @default(cuid())
  extracurricularId String
  title             String
  date              DateTime
  result            String?
```

```
}

// ===== BK / KONSELING =====
model CounselingTicket {
  id         String  @id @default(cuid())
  studentId  String
  createdBy  String
  topic      String
  status     String   @default("open") // open/in_progress/closed
  priority   String?  // low/med/high
  sessions   CounselingSession[]
}

model CounselingSession {
  id        String  @id @default(cuid())
  ticketId  String
  counselorId String
  sessionDate DateTime
  notesPriv  String?
  ticket    CounselingTicket @relation(fields: [ticketId], references: [id])
}

model CounselingRef {
  id        String  @id @default(cuid())
  ticketId  String
  referredTo String
  notes     String?
  ticket    CounselingTicket @relation(fields: [ticketId], references: [id])
}

// ===== CBT / LMS =====
model LmsLink {
  id         String @id @default(cuid())
  provider   String // moodle/google/other
  courseId   String?
  classId    String?
  subjectId  String?
  syncPolicy String?
}

model LmsScore {
  id       String @id @default(cuid())
  linkId   String
  studentId String
  score    Float
  syncedAt DateTime @default(now())
}

model Exam {
  id        String @id @default(cuid())
  title     String
```

```
  subjectId String?
  classId   String?
  startAt   DateTime?
  endAt     DateTime?
  questions ExamQuestion[]
  attempts  ExamAttempt[]
}

model ExamQuestion {
  id       String  @id @default(cuid())
  examId   String
  text     String
  options  Json?
  answer   Json?
  exam     Exam    @relation(fields: [examId], references: [id])
}

model ExamAttempt {
  id        String  @id @default(cuid())
  examId    String
  studentId String
  answers   Json?
  score     Float?
  startedAt DateTime @default(now())
  submittedAt DateTime?
  exam      Exam    @relation(fields: [examId], references: [id])
  @@unique([examId, studentId])
}

// ===== ANALITIK & NOTIFIKASI =====
model AuditEvent {
  id        String   @id @default(cuid())
  actorId   String?
  type      String
  entity    String?
  entityId  String?
  meta      Json?
  occurredAt DateTime @default(now())
  @@index([type, occurredAt])
}

model WaTemplate {
  id       String @id @default(cuid())
  key      String @unique
  content  String
  variables Json?
}

model WaOutbox {
  id         String  @id @default(cuid())
  templateId String?
```

```
  to         String
  payload    Json?
  status     WaStatus @default(PENDING)
  providerMsgId String?
  sentAt     DateTime?
  template   WaTemplate? @relation(fields: [templateId], references: [id])
  @@index([status, sentAt])
}

// ===== CMS =====
model CmsPost {
  id          String  @id @default(cuid())
  title       String
  slug        String  @unique
  content     String
  authorId    String?
  publishedAt DateTime?
}

model CmsEvent {
  id        String  @id @default(cuid())
  title     String
  date      DateTime
  location  String?
  description String?
}
```

> **Catatan penting:** - Beberapa aturan bisnis (mis. saldo tabungan tidak boleh negatif, agregasi raport) ditegakkan di layer service/transaction; tambahkan constraint DB lewat migration SQL bila diperlukan. - Tambahkan index tambahan sesuai kebutuhan query aktual.

12.1 Kamus Data (ringkas untuk tabel kritikal)

- **Invoice/InvoiceItem/Payment**: total invoice = sum(items.amount); status berpindah ke `PARTIAL/PAID` berdasar total payment.
- **SavingsAccount/SavingsTransaction**: saldo = sum(DEPOSIT) − sum(WITHDRAW) ± ADJUSTMENT; `WITHDRAW` wajib cek saldo cukup.
- **Attendance/AttendanceStaff**: status dihitung dari schedule/shift + grace. Koreksi disimpan sebagai update dengan audit event.
- **ReportCard**: 1 per siswa per semester (unique) + `publishedAt` saat tampil di portal orang tua.
- **LibLoan**: denda = max(0, hari_terlambat × `finePerDay`). Pedoman Implementasi (untuk Codex/VSCode)
- Stack: Next.js, Prisma, NextAuth, Zod, RHF, React Query, Tailwind, shadcn/ui.
- Struktur:

```
src/
  app/(public|auth|dashboard)/*
  app/api/*
  components/*
```

```
  lib/{auth,db,rbac,validators,wa,analytics}.ts
  features/
{attendance,hr,assessments,report,ppdb,finance,savings,library,assets,extras,counseling,l
*
prisma/schema.prisma
prisma/seed.ts
```

- Konvensi: kebab-case file, camelCase var/fungsi, PascalCase komponen.
- Commit: Conventional Commits.
- Validasi: Zod.
- RBAC: middleware guard.
- Transaksi: prisma.$transaction untuk keuangan/tabungan.
- WA: sendWA(templateKey, to, vars) → enqueue wa_outbox → worker kirim.
- Testing: unit, integration, e2e (Playwright).
- Dokumentasi: generate OpenAPI dari Zod.
- TODO markers untuk milestone.
