Admin Phase 2 Readiness Audit Report

Date: 2025-12-21 **Auditor:** Antigravity (Senior System Auditor) **Context:** SIXKUL Project - Admin Phase 2

---

1) Attendance Data Integrity

Question: Are ALL attendance records strictly tied to a Session entity? Is there ANY remaining legacy schedule-based attendance logic or data?

Answer: NO

Evidence:

- **Schema Permissiveness:** `prisma/schema.prisma` defines `session_id` on the `Attendance` model as `String?` (optional), physically allowing orphans.
- **API Looseness:** `src/app/api/attendance/batch/route.ts` accepts an optional `scheduleId` payload and maps it to `session_id: scheduleId || null`. This allows creating attendance records with `session_id: null` if the client omits the field.
- **Legacy Naming Leak:** The variable name `scheduleId` in `batch/route.ts` (used to populate `session_id`) indicates a semantic leak from legacy schedule-based logic.

---

2) Session Immutability

Question: Once attendance exists for a session, is that session prevented (by code or logic) from being deleted or modified?

Answer: YES

Evidence:

- **Deletion Guard:** `src/lib/pembina-session-data.ts` exports `deleteSession`, which calls `canDeleteSession(sessionId)`. This function explicitly checks `prisma.attendance.count({ where: { session_id: sessionId } })`. If count > 0, strict deletion prevention is enforced.
- **Modification Prevention:** There is **NO** code in the codebase (verified via global grep for `updateSession`, `prisma.session.update`) that allows updating a Session entity. Immutability is enforced by the absence of mutator functions.

---

3) Attendance Mutability

Question: Can attendance records be edited after initial submission? If YES, under what conditions?

Answer: YES

Evidence:

- **Upsert Logic:** Both `src/lib/pembina-attendance-data.ts` (`saveSessionAttendance`) and `src/app/api/attendance/batch/route.ts` utilize `prisma.$transaction` with `upsert` operations.
- **Conditions:** A PEMBINA can overwrite existing attendance statuses (e.g., from PRESENT to SICK) at any time by submitting a new request for the same `enrollment_id` and `date`. There are no temporal locking mechanisms (e.g., "lock after 24h") found in the code.

---

4) Enrollment State Correctness

Question: Are enrollment states strictly enforced as: PENDING → ACTIVE → REJECTED? Is attendance creation blocked for non-ACTIVE enrollments?

Answer: YES

Evidence:

- **Attendance Blocking:** `src/lib/pembina-attendance-data.ts` -> `getEnrollmentsForAttendance` explicitly includes `where: { status: "ACTIVE" }`. Non-active enrollments are never retrieved for attendance input, effectively blocking creation.
- **State Definition:** `prisma/schema.prisma` defines the `EnrollmentStatus` enum (`PENDING`, `ACTIVE`, `REJECTED`, etc.), and `src/lib/attendance-data.ts` enforces interactions only on valid states.

---

5) Timestamp & Timezone Sanity

Question: Are timestamps present and consistent for session creation, attendance records, and enrollment creation? Are they stored in UTC/normalized?

Answer: YES

Evidence:

- **Schema Standards:** `prisma/schema.prisma` includes `created_at DateTime @default(now())` and `updated_at` on strictly all relevant models (`Session`, `Attendance`, `Enrollment`, `User`).
- **Handling:** `src/lib/pembina-session-data.ts` utilizes `date-fns` (`startOfDay`, `endOfDay`, `parseISO`) to normalize dates before database queries, ensuring consistency. Prisma/PostgreSQL default behavior handles UTC storage.

---

6) Soft-Delete Consistency

Question: Are deletes implemented as soft-deletes for users, extracurriculars, sessions? Is historical data preserved?

Answer: NO

Evidence:

- **Hard Deletes:** `src/lib/pembina-session-data.ts` uses `prisma.session.delete(...)`.
- **Schema:** `prisma/schema.prisma` relations use `onDelete: Cascade` (e.g., User → StudentProfile).
- **Missing Columns:** There are **NO** `deleted_at` or `is_deleted` fields in the `prisma/schema.prisma` for `User`, `Extracurricular`, or `Session` models. Deletion is destructive and permanent.

---

7) Cardinality Guarantees

Question: confirm the following:

- Can an extracurricular have multiple PEMBINA?
- Can a PEMBINA manage multiple extracurriculars?
- Can a student have multiple ACTIVE enrollments?

Answer:

- **Extracurricular have multiple PEMBINA? NO.**

  - **Evidence:** `prisma/schema.prisma -> Extracurricular` model has `pembina_id String` (Scalar). strictly 1-to-1 mapping from Extracurricular to Pembina.
- **PEMBINA manage multiple extracurriculars? YES.**
  - **Evidence:** `prisma/schema.prisma -> PembinaProfile` model has `extracurriculars Extracurricular[]`. One Pembina can be assigned to multiple activities.
- **Student have multiple ACTIVE enrollments? YES.**
  - **Evidence:** `prisma/schema.prisma` -> `StudentProfile` has `Enrollment[]`. The only constraint is `@@unique([student_id, extracurricular_id])`, which prevents double-enrollment in the *same* activity, but allows concurrent active enrollments in *different* activities.

---

8) Existing Aggregation Logic

Question: Does the codebase already contain ANY aggregation queries (counts, summaries, grouped queries) intended for dashboards?

Answer: YES

Evidence:

- **Dashboard Data:** `src/lib/admin-dashboard-data.ts` explicitly calls `prisma.enrollment.groupBy({ by: ["status"], _count: true })` to calculate statistics.
- **Session Counts:** `src/lib/pembina-session-data.ts` uses `_count: { attendances: true }` in `include` clauses to summarize attendance data per session.