AUB CS Capacity-Constraint System

Start Date: September 6, 2025 (Beirut)

---

1) Purpose & Problem Statement

Goal: Replace the current manual Google Forms → Google Sheets → Chair review pipeline with a robust, auditable system that enforces academic rules and capacity policies at the **data layer**, exposes a **student submission UI**, and a **chair/department UI** for triage, review, and decisioning.

Current issues:

- Fragmented data capture; heavy manual verification and filtering by the Chair.
- Weak validation (students can misreport; multiple conflicting requests possible).
- No strong audit trail, deduplication, or constraint enforcement.
- Slow turnaround; difficult bulk actions and prioritization.

High-level success criteria:

- Accurate, validated requests at submission time; minimized manual checks.
- Deterministic, transparent enforcement of prerequisites/capacity/eligibility.
- One consolidated database as **source of truth** with auditability.
- Efficient chair workflows (search, filter, bulk approve/reject, notes).

---

2) Stakeholders & Roles

- **Students**: Submit capacity/override requests, upload transcript evidence.
- **Chair / Department reviewer(s)**: Review, comment, approve/reject/waitlist; apply policies.
- **GAs/Engineering team**: Design schema, implement validation & workflows, maintain system.
- **Registrar/Advising (future)**: Potential integration (read-only) or policy inputs.

---

3) Current Flow (As-Is)

1. Student fills Google Form.
2. Responses land in Google Sheet.
3. Chair scans rows, validates manually (prereqs, duplicates, necessity, etc.).
4. Chair emails decisions or adjusts caps manually via SIS (outside the sheet).

Pain points: minimal validation, duplicated/conflicting requests, no dedupe per student+course+term, hard to prove eligibility or necessity, time-consuming.

---

4) Target System (To-Be) Overview

Two Interfaces:

- **Student portal** (submit request + evidence, track status).
- **Chair portal** (queue of requests, powerful filters, bulk actions, decision logging).

Database-centric design:

- All entities (students, courses, sections, prerequisites, requests, decisions, transcripts) live in a relational database (SQL likely).
- **Data invariants** (uniqueness, prerequisite satisfaction, per-term limits) enforced by keys, constraints, and transactional checks.

---

5) Student Submission — Minimum Data Set

- **Identity**: Banner ID, AUB email (SSO in future), name (read-only from directory if integrated).
- **Academic context**: Major, cohort year, program (BS/BE), expected graduation term (optional), advisor (optional).
- **Term**: e.g., `Fall 2025`.
- **Request type** (categorize for policy):
  - **Capacity override** (section full)
  - **Prerequisite override** (missing prereq, special permission)
  - **Overload** (exceeding credit limit)
  - **Time-conflict exception** (optional, future)
- **Course/Section**: Course code (e.g., CMPS 244), preferred section(s) if any.

- **Reason/Justification**: Free text + dropdown reasons (major requirement, graduation-urgent, sequence dependency, etc.).
- **Evidence**: **Transcript upload** (required for certain request types), optional supporting docs.
- **Attestations**: e.g., “I have not submitted more requests than allowed”, “information is accurate”.

---

6) Chair / Department UI — Minimum Capabilities

- **Queue**: Pending requests with sortable columns and filters (term, course, reason, priority, status).
- **Search**: Banner ID, name, course code; quick student history view.
- **Detail page**: Request metadata, transcript preview, prerequisite/eligibility assessment, prior decisions.
- **Decisioning**: Approve / Reject / Waitlist; add internal/external notes; one decision per request; timestamped.
- **Bulk actions**: Approve batches (e.g., all seniors for course X); export CSV.
- **Audit**: Immutable log entries for status changes, who did what, when.

---

7) Core Entities (Initial Draft)

- **STUDENT**: `id`, `banner_id`, `major`, `cohort_year`.
- **COURSE**: `code`, `title`, `credits`, `department`.
- **SECTION**: `id`, `course_code` → `COURSE`, `term`, `capacity`.
- **PREREQ**: `(course_code, requires_code, type={hard, soft, co})`.
- **TRANSCRIPT_ENTRY**: `(student_id, course_code, term, grade)`.
- **CAPACITY_REQUEST**: `(id, student_id, term, course_code, section_optional, type, reason, status, created_at)`.
- **DECISION**: `(id, request_id UNIQUE, reviewer_id, outcome={approve,reject,waitlist}, note, decided_at)`.
- **TERM_POLICY** (future): per-term caps (e.g., max elective requests per student).

---

8) Immediate Next Steps

1. **Draft ER Diagram** (Mermaid + notes) for review — include entities above and FK/unique/partial-index ideas.
2. **Collect sample fields from current Google Sheet** used by the Chair; map to proposed schema.