Frontend (Portal) — What to build

Roles & screens

Admin

- Upload Wizard (CSV/Excel bundle)
- Master Data pages (Sections, Rooms, Faculty, Courses)
- Generate Draft Timetable (run solver)
- Publish versions (v1, v2…) + rollback
- Policies (constraints + priorities)

HOD

- View section/faculty/room timetable
- Conflict panel + fix suggestions
- Approve repair suggestions
- Publish request (optional)

Class Advisor

- Section timetable editing (drag-drop)
- Raise change request (substitution / room change)

Faculty

- Personal schedule view (today + week)
- Notifications + change log

Frontend stack (recommended)

- **Next.js + TypeScript**
- **Auth**: NextAuth (JWT) or custom
- **UI**: Tailwind + shadcn
- **Timetable UI**: grid + drag/drop (react-beautiful-dnd / dnd-kit)
- **State**: Zustand/Redux
- **API**: REST to FastAPI (or GraphQL later)

Backend — What to build

Core services/modules

1. **Ingestion Service**
    - Upload bundle
    - Validate + normalize (IDs, naming)
    - Store canonical tables (Postgres)
2. **Scheduling Service (Solver)**
    - OR-Tools CP-SAT

    - Produce draft timetable + metrics + conflicts
3. **Conflict & Repair Service**
    - Detect conflicts after manual edits/changes
    - “Incremental repair”: freeze most timetable and re-solve only impacted parts
4. **Versioning & Audit**
    - Each publish = new version
    - Diff between versions
    - Rollback
5. **Notifications Service**
    - Email/WhatsApp/in-app
    - “Your class moved to AB-102 Tue P3”

Backend stack

- **FastAPI**
- **Postgres - Google Cloud SQL*
- **Redis + Celery/RQ** (async generation/repair jobs)
- **OR-Tools** (solver)
- Optional later: **WebSockets** for live updates

AI Model / Agent Layer — What it does (correctly)

Important: **LLM doesn’t create the timetable.** Solver does.

Agent responsibilities

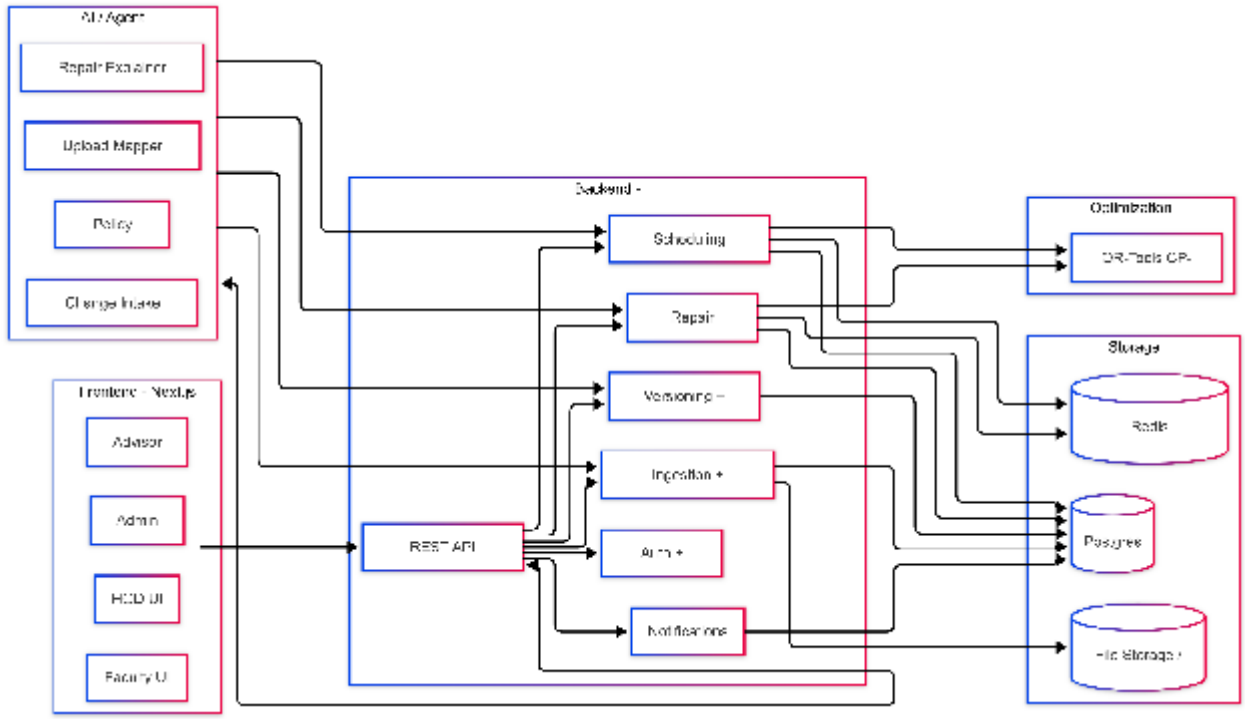
1. **Upload Mapper Agent**
    - “DBMS Lab” = “Database Lab”
    - Fix column mismatch, detect missing info, ask user questions
2. **Policy Agent**
    - Converts HOD policy into solver-friendly config (priorities)
3. **Repair Explainer Agent**
    - “Why did we move OS from Wed P2 to Wed P4?”
    - Produces human-readable explanation + impact list
4. **Change Intake Agent**
    - Natural language: “Swap CSE2A Tue P2 with Wed P5”
    - Converts to structured change request

Model choices (practical)

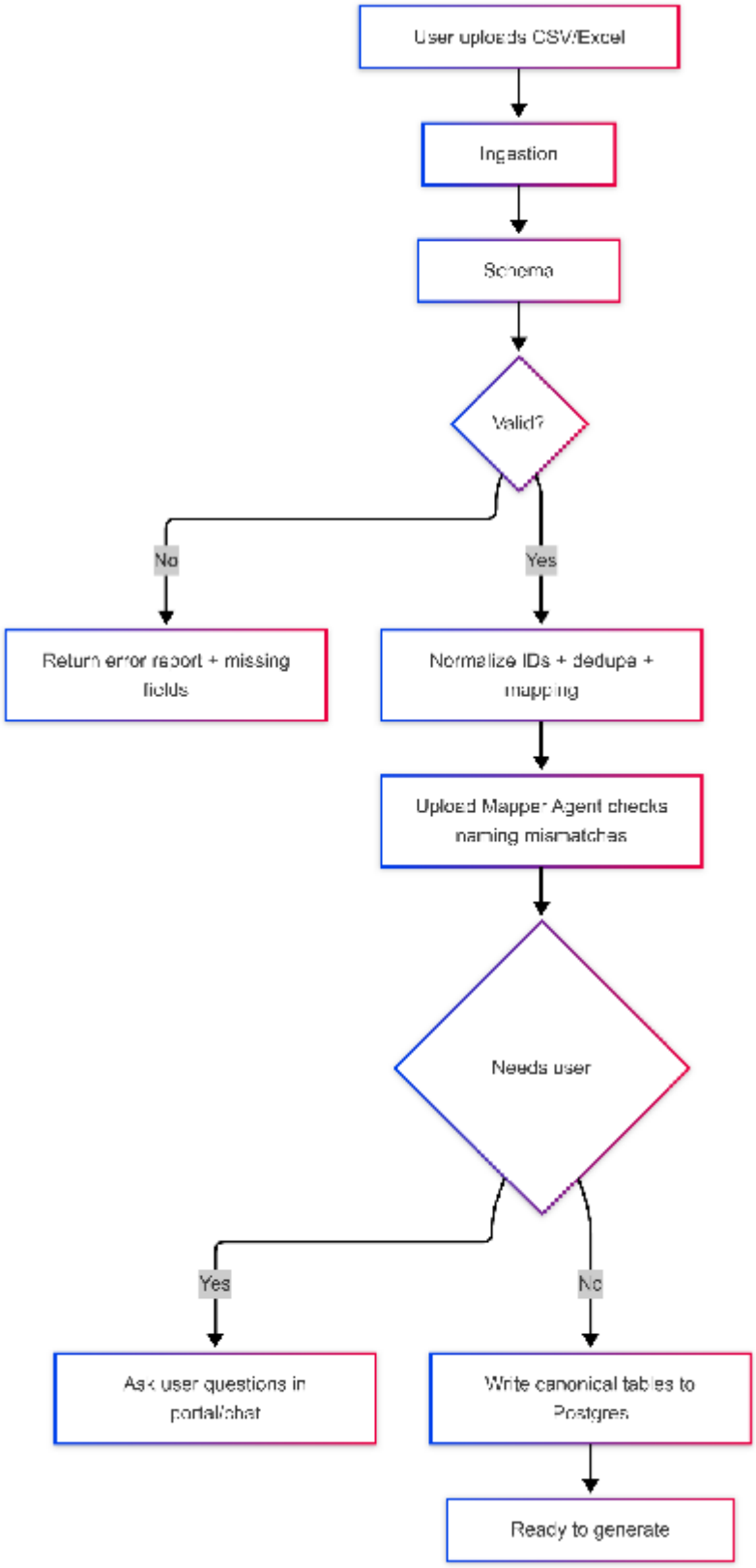
- LLM: Gemini (your preference)
- Embeddings (optional): store faculty/course synonyms
- Store agent traces (for audits)

---

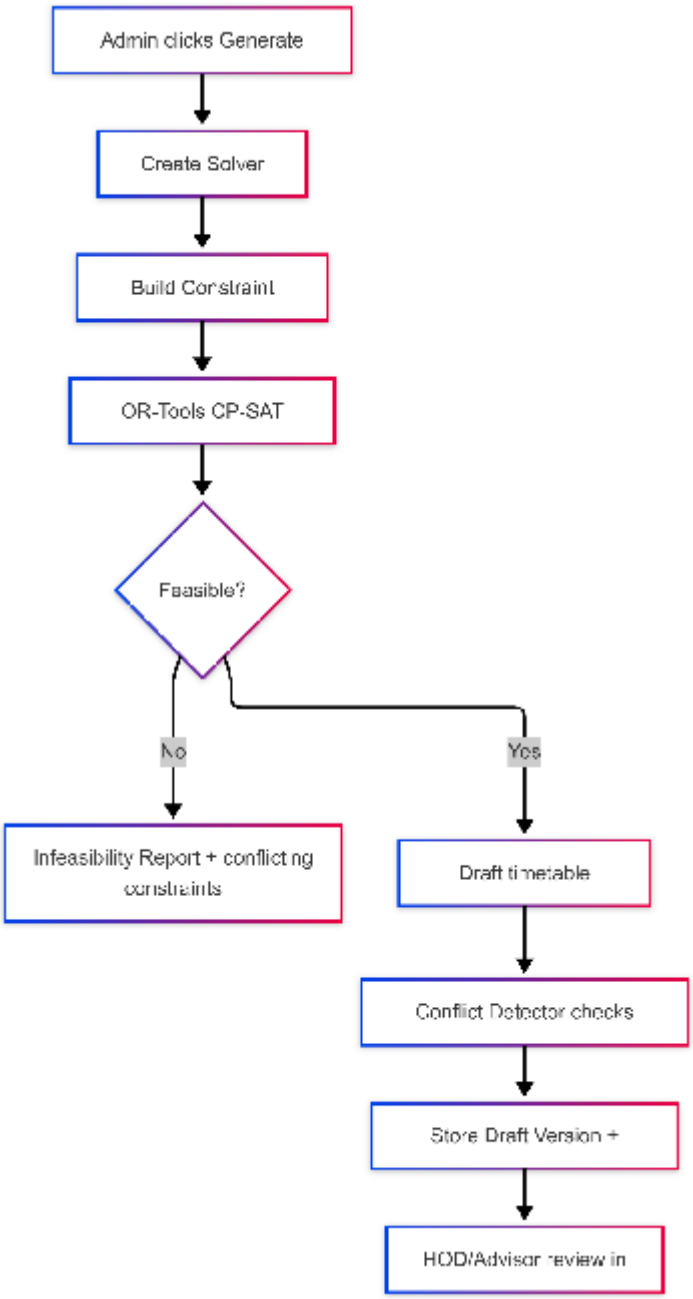
1) System Architecture (Frontend + Backend + AI + Solver)



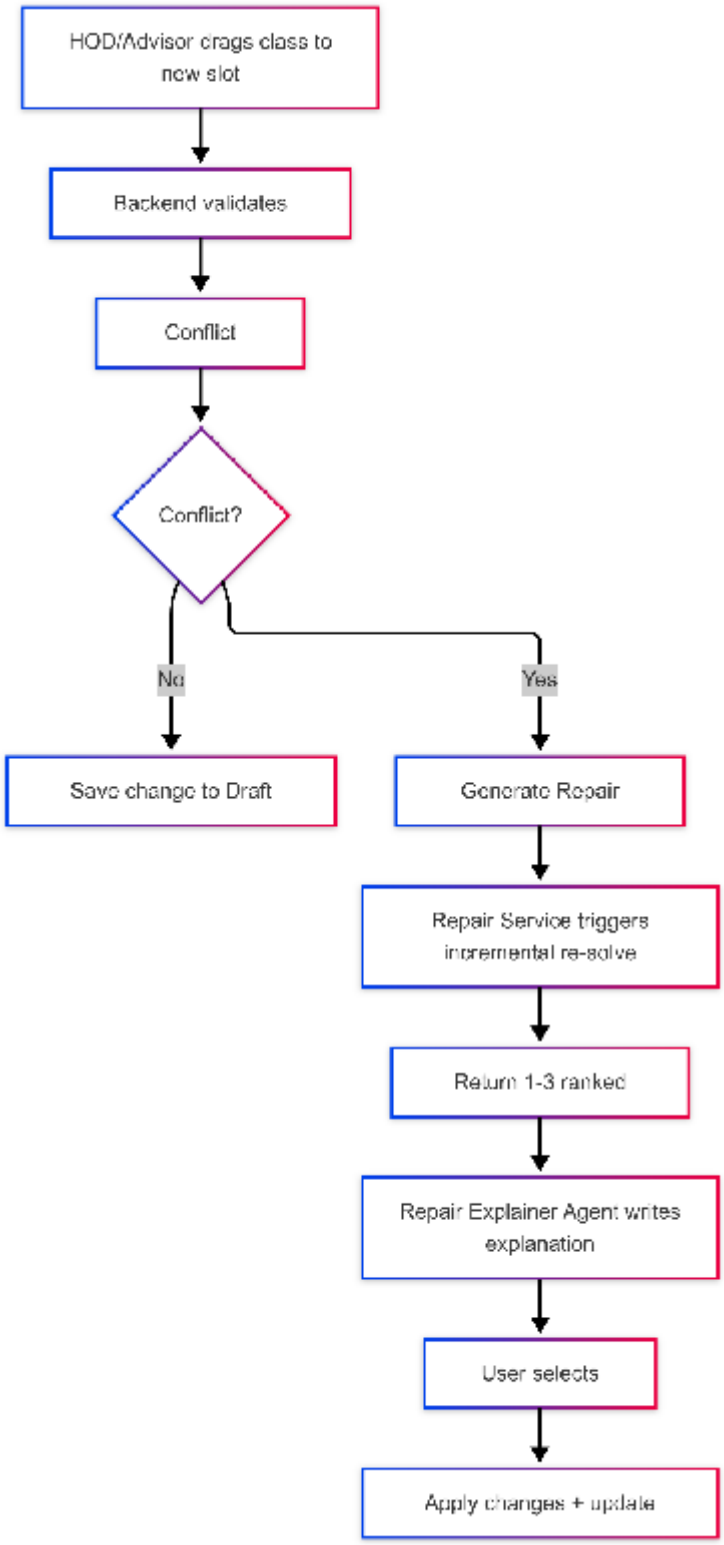
2) Upload → Normalize → Validate Workflow



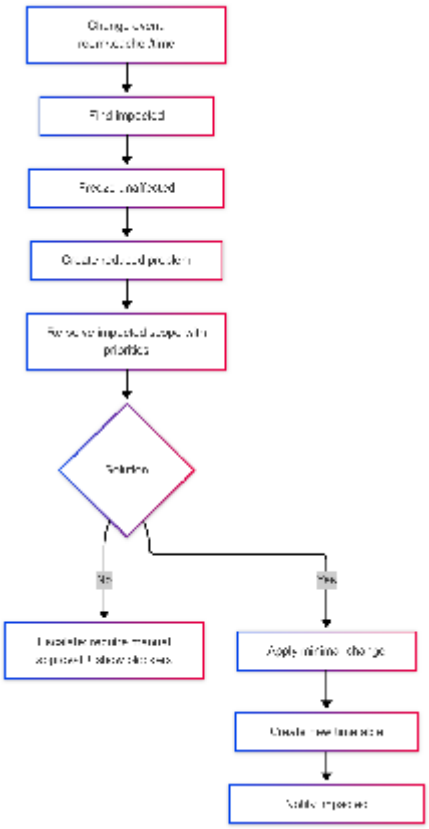
3) Generate Draft Timetable (Solver Job)



4) Manual Edit → Conflict Detection → Repair Options



5) Auto-Repair Logic (Your “sequential handling”)



6) Publish + Notifications