CodoraAI Timetable Agent

Complete End-to-End Development Roadmap

BIG PICTURE (Keep This Fixed)

Your product has **4 brains**, built **in this exact order**:

1️⃣ **Constraint Brain** (Solver)
2️⃣ **Repair Brain** (Incremental fixing)
3️⃣ **Trust Brain** (Explain + approve)
4️⃣ **Convenience Brain** (AI + UI)

⚠ **Break this order → product fails.**

PHASE 0 — Product Definition (Day 0–1)

Objective

Freeze scope so you don't keep changing direction.

Define This Clearly (Write It Down)

Target:
Engineering colleges (India)

Users:

- Admin (Academic office)
- HOD
- Class Advisor
- Faculty

Core Promise:

> "Conflict-free timetable + safe auto-repair with explanations"

❌ Do **NOT** promise "AI timetable"
☑ Promise **"error-free + explainable + change-friendly timetable"**

PHASE 1 — Data Foundation (Week 1)

Goal

Make data predictable, even if uploads are messy.

1️⃣ Finalize Input Contracts (**MOST IMPORTANT**)

Lock these templates **forever**:

- `sections.csv`
- `rooms.csv`
- `faculty.csv`
- `courses.csv`
- `faculty_course_map.csv`
- `time_config.json`

For **each column**, define:

- Meaning
- Allowed values
- Example

👉 This is your **API contract with colleges**.
⚠️ If this is weak → everything breaks.

2️⃣ Validation Engine

Build **strict validation**:

- Missing columns
- Wrong room types
- Capacity mismatch
- Unknown faculty/course/section
- Impossible constraints (e.g., lab with no lab room)

```
{
  "errors": [],
  "warnings": [],
  "suggestions": []
}
```

3️⃣ Normalization Layer

Before solver:

- Trim spaces
- Unify naming
- Generate internal IDs

🚫 This is **non-negotiable**.

PHASE 2 — Core Timetable Solver (Week 2)

Goal

Generate a **guaranteed valid timetable**.

4 Hard Constraints (Never Violated)

- One class per section per slot
- One class per room per slot
- One class per faculty per slot
- Room capacity ≥ section strength
- Room type matches course type
- Faculty max load per day
- Shift timing respected

---

5 Medium Constraints

- Labs must be consecutive (2–3 slots)
- Avoid faculty having 4 continuous periods
- No class during recess

---

6 Soft Optimization (Later)

- Minimize faculty idle gaps
- Minimize room changes
- Spread heavy subjects

PHASE 3 — Repair & Change Handling (Week 3)

7 Conflict Detector

Detect:

- Room conflict
- Faculty conflict
- Section conflict
- Capacity issue

---

8 Incremental Repair Engine (**CRITICAL**)

- Identify impacted nodes only
- Freeze everything else
- Re-solve small sub-problem
- Rank by minimum change

9️⃣ Versioning System

Track:

- Who changed
- What changed
- Why changed

PHASE 4 — Minimal Frontend (Week 4)

1️⃣0️⃣ Upload Wizard

1️⃣1️⃣ Timetable Views

1️⃣2️⃣ Manual Change Screen

PHASE 5 — AI Agents (Week 5)

1️⃣3️⃣ Upload Mapper Agent

1️⃣4️⃣ Repair Explainer Agent

1️⃣5️⃣ Natural Language Change Agent

PHASE 6 — Notifications & Roles (Week 6)

1️⃣6️⃣ Role-Based Access

1️⃣7️⃣ Notifications

PHASE 7 — Production Hardening (Week 7)

1️⃣8️⃣ Performance

1️⃣9️⃣ Data Safety

2️⃣0️⃣ Pilot Deployment

FINAL PRODUCT STACK

Frontend → FastAPI → Solver → AI Agents → DB → Notifications

MOST IMPORTANT ADVICE

❌ Don’t start with AI
❌ Don’t start with UI

☑ Start with constraints + repair
☑ Make failures explainable