Our goals are:

- **10 entities** in your ER model
- **10 relationships**
- **12 total system operations** spanning **Members + Trainers + Admins**
- **1 View, 1 Trigger, 1 Index**

TECH STACK

Python + FASTAPI + PostgreSQL + psycopg2 or SQLAlchemy

Entities and Attributes

Member

- **member_id** (PK)
- name
- email (**unique**)
- date_of_birth
- gender
- phone
- created_at (timestamp, default now())

Trainer

- **trainer_id** (PK)
- name
- email (**unique**)
- specialization
- phone

AdminStaff

- **admin_id** (PK)
- name
- email (**unique**)
- role

FitnessGoal

- **goal_id** (PK)
- member_id (FK → Member)
- goal_type (e.g., “Lose Weight”)
- target_value (numeric value)
- is_active (boolean, default true)
- created_at (timestamp)

HealthMetric

- **metric_id** (PK)
- member_id (FK → Member)
- weight (numeric)
- heart_rate (integer)
- body_fat (numeric)
- recorded_at (timestamp) *(each entry is historical, no overwrite)*

PersonalTrainingSession

- **session_id** (PK)
- member_id (FK → Member)

- trainer_id (FK → Trainer)
- room_id (FK → Room)
- start_time (timestamp)
- end_time (timestamp)
- status (scheduled / cancelled / completed)

GroupClass

- **class_id** (PK)
- class_name
- trainer_id (FK → Trainer)
- room_id (FK → Room)
- start_time (timestamp)
- end_time (timestamp)
- capacity (integer)

ClassRegistration *(Member* ↔ *GroupClass)*

- **registration_id** (PK)
- member_id (FK → Member)
- class_id (FK → GroupClass)
- registered_at (timestamp)

Room

- **room_id** (PK)
- room_name
- location
- capacity (integer)

Equipment

- **equipment_id** (PK)
- room_id (FK → Room)
- name
- status (working / broken / in_repair)

TrainerAvailability

- **availability_id** (PK)
- trainer_id (FK → Trainer)
- start_time (timestamp)
- end_time (timestamp)

MaintenanceRecord

- **maintenance_id** (PK)
- equipment_id (FK → Equipment)
- issue_description (text)
- reported_at (timestamp)
- resolved_at (timestamp, *nullable*)
- status (open / in_progress / resolved)

Operations:
2.1 Member Functions
Members interact with the system to manage personal data, monitor progress, and schedule services..
● User Registration: Create a new member with unique email and basic profile info.
● Profile Management: Update personal details, fitness goals (e.g., weight target), and input new
health metrics (e.g., weight, heart rate).
● Health History: Log multiple metric entries; do not overwrite. Must support time-stamped entries.
● Dashboard: Show latest health stats, active goals, past class count, and upcoming sessions.
● Group Class Registration: Register for scheduled classes if capacity permits.

2.2 Trainer Functions
Trainers manage their working hours and access limited member data. At least 2 operations must
be implemented.
● Set Availability: Define time windows when available for sessions or classes. Prevent overlap.
● Schedule View: See assigned PT sessions and classes.
● Member Lookup: Search by name (case-insensitive) and view current goal and last metric. No editing rights.

2.3 Administrative Staff Functions
Admins coordinate resources and simulate payments. At least 2 operations must be implemented.
● Room Booking: Assign rooms for sessions or classes. Prevent double-booking.
● Equipment Maintenance: Log issues, track repair status, associate with room/equipment.
● Class Management: Define new classes, assign trainers/rooms/time, update schedules.
● Billing & Payment: Generate bills, add line items, record payments. Simulate status updates

Short Data Flow Explanation

When a user performs an action (e.g., register member):

1. **Client / Tester uses FastAPI Swagger UI** `/docs`
   → sends a JSON request.

2. **Schema (Pydantic)**
   → validates and structures the input data.

3. **Router Endpoint**
   → receives schema data → calls service/database logic.

4. **Database Session (SQLAlchemy)**
   → converts schema data into a **Model object**
   → executes SQL to insert/update/select.

5. **Database (PostgreSQL)**
   → stores the actual data.

6. **Response Schema** (optional)
   → turns database objects back into clean API output.