Design Document: Physician Appointment Scheduler

---

1. Entity Relationship Diagram (ERD)

The following tables are used:

- **physician** (id, name, specialization, clinicId)
- **patient** (id, name, contact)
- **clinic** (id, name, location)
- **availability** (id, date, start_time, end_time, physicianId, clinicId)
- **appointment** (id, start_time, end_time, status, physicianId, patientId, clinicId)
- **billing_rule** (id, gap_before_minutes, gap_after_minutes, min_gap_between_appointments, physicianId)

Relationships:

- One appointment is linked to one physician, one patient, and one clinic.
- One availability block is defined for one physician and clinic.
- One billing_rule is defined per physician.

---

2. Key API

POST /api/appointments/recommend

Suggests top 10 appointment slots for a patient and physician on a preferred date.

Input JSON:

```
{
  "clinicId": "c001",
  "physicianId": "p001",
  "patientId": "u123",
  "preferredDate": "2025-07-01",
  "durationMinutes": 15
}
```

Output JSON:

```
{
  "status": "success",
  "recommendedSlots": [
    "2025-07-01T09:00:00",
    "2025-07-01T10:00:00",
    ...
  ]
}
```

Note: There is no use of patient id in this api so it’s just added as required without any use.

---

3. Scheduling Algorithm Logic Flow (Slot Recommendation Algorithm)

1. **Fetch Availability:**
   - Load physician’s availability for the given date.
   - If no availability, return an empty list.
2. **Fetch Billing Rules:**
   - Load physician’s billing rules (gap_before, gap_after, min_gap_between_appointments).
3. **Fetch Existing Appointments:**
   - Load physician’s appointments for that date.
   - Apply buffer zones to all appointments using gap_before and gap_after.
4. **Loop Through Availability Block:**
   - Starting from availability start time.
   - Try each slot = current_time + duration.
   - Check for overlap with any appointment (including buffer zones).
5. **Slot Evaluation:**
   - If valid (no overlap):
     - Calculate disruption score = total distance to other appointments.
     - Add to the candidate list.
     - Jump ahead by (duration + min_gap_between_appointments).
   - If invalid (overlaps):
     - Skip to end of conflict + gaps.
6. **Return Results:**
   - Sort valid slots by disruption score (least disruptive first).
   - Return top 10 as recommendedSlots.

---

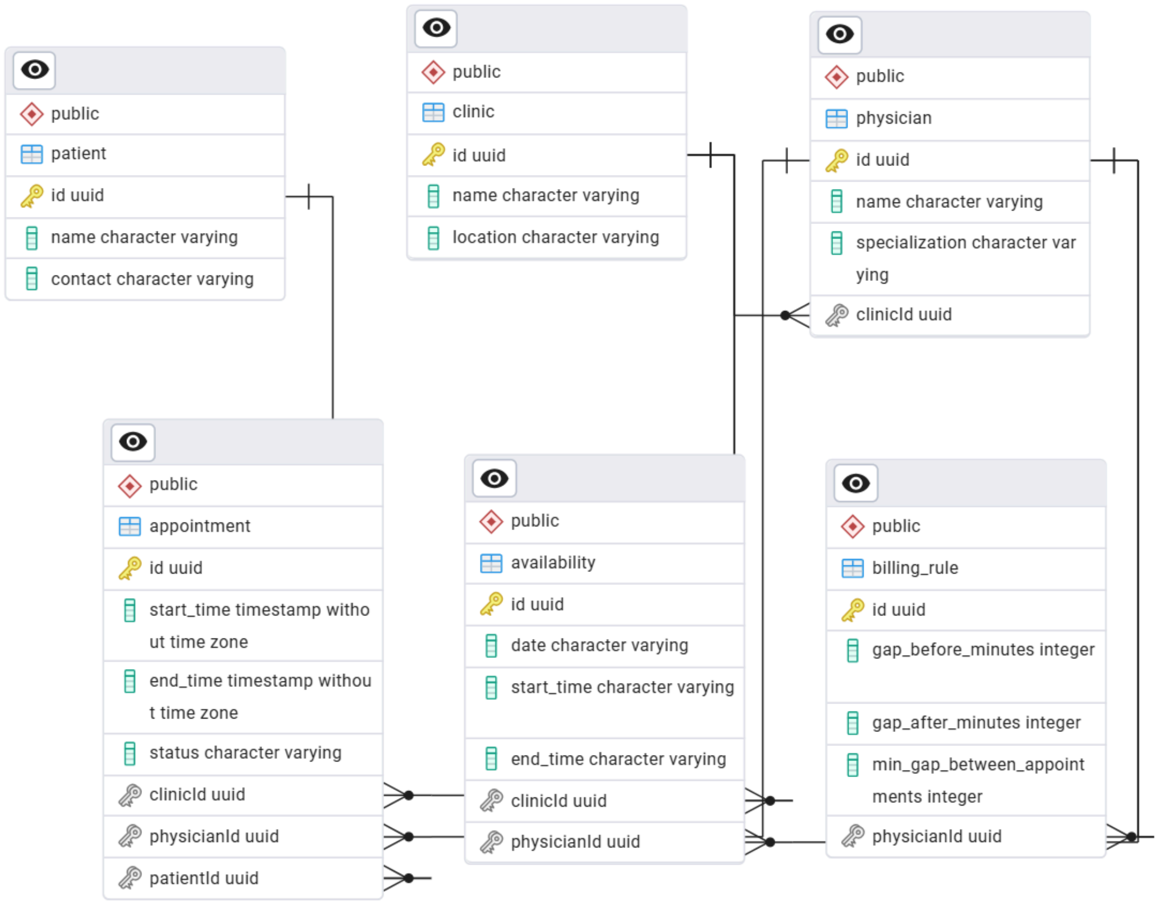
4. How the System Handles Gaps and Recommendations

- **Gap Before/After:** Applied as buffer time around each appointment to avoid back-to-back bookings.
- **Min Gap Between Appointments:** Ensures future slots aren't too close to the last scheduled appointment.
- **Disruption Score:** Measures time difference between midpoint of new slot and existing appointments to avoid clustering.
- **Efficiency:** Algorithm loops only within the available window, avoids unnecessary iterations by skipping blocked periods.

Notes

- Code is implemented as a modular NestJS service (`SchedulerService`).
- Can easily be extended to support custom rules per clinic.
- Designed for scalability with minimal table joins and clear separation of concerns.

ERD Diagram:


public
patient
id uuid
name character varying
contact character varying
public
clinic
id uuid
name character varying
location character varying
public
physician
id uuid
name character varying
specialization character varying
clinicId uuid
public
appointment
id uuid
start_time timestamp without time zone
end_time timestamp without time zone
status character varying
clinicId uuid
physicianId uuid
patientId uuid
public
availability
id uuid
date character varying
start_time character varying
end_time character varying
clinicId uuid
physicianId uuid
public
billing_rule
id uuid
gap_before_minutes integer
gap_after_minutes integer
min_gap_between_appointments integer
physicianId uuid