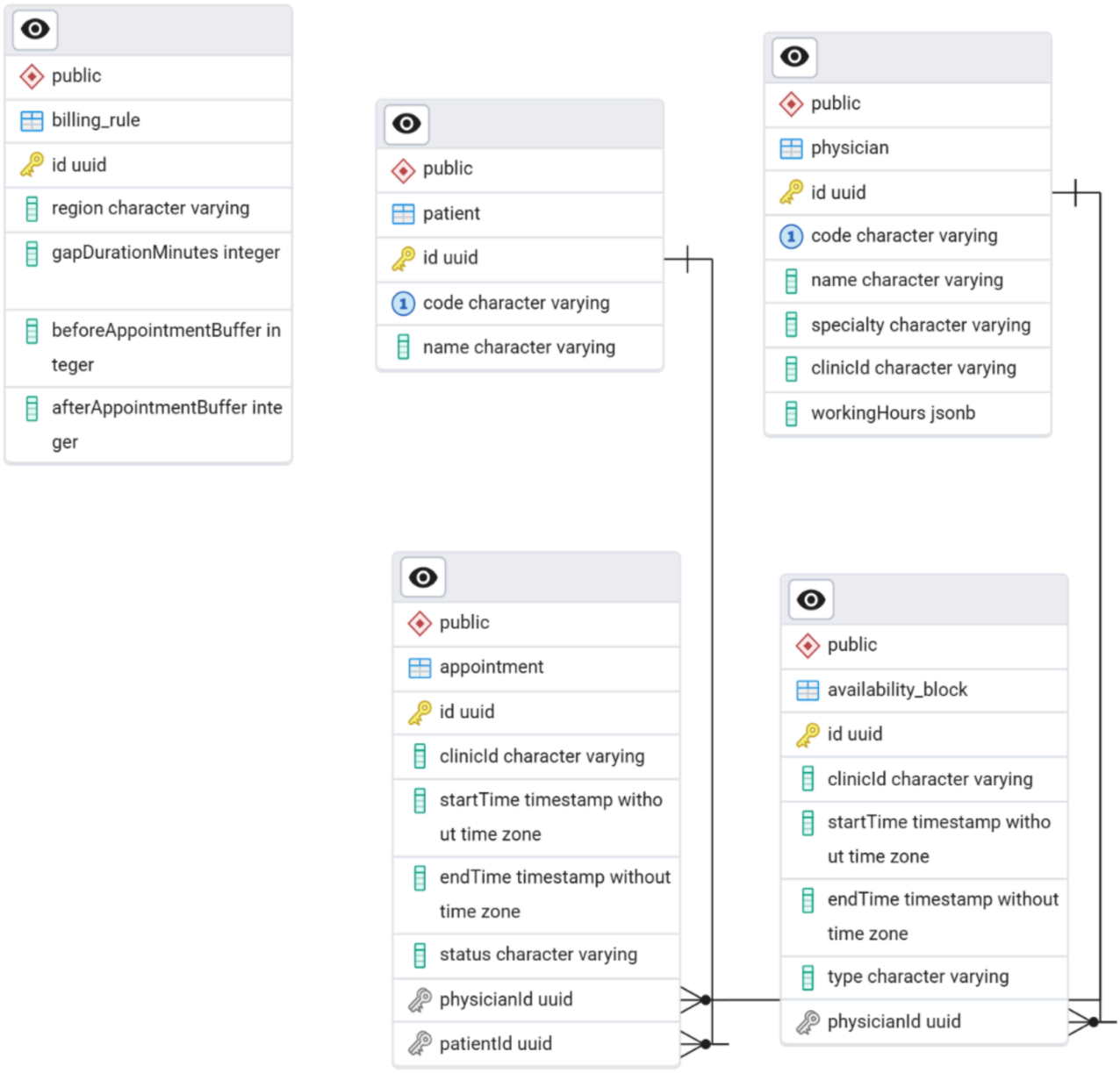
Nextra EHR Scheduling Backend – Design & Submission Report
Prepared by: Neharika Puri

This report details the design and implementation of the Nextera EHR scheduling module backend, built with NestJS and PostgreSQL. It outlines the system's architecture, key API endpoints, the core scheduling algorithm, and underlying assumptions, ensuring a comprehensive understanding of the solution.

1. ERD (Entity Relationship Diagram)

The Entity Relationship Diagram (ERD) visually represents the structure of the database and the relationships between its core entities.



Entities:

- **Physician**: Represents healthcare providers.
  - id (UUID): Unique identifier for the physician.
  - code (Character Varying): A short code for the physician.
  - name (Character Varying): Full name of the physician.
  - specialty (Character Varying): Physician's medical specialty.
  - clinicId (Character Varying): Identifier for the clinic the physician is associated with.
  - workingHours (JSONB): Stores the physician's working hours, typically by day of the week.
- **Patient**: Represents individuals seeking appointments.
  - id (UUID): Unique identifier for the patient.
  - code (Character Varying): A short code for the patient.
  - name (Character Varying): Full name of the patient.
- **Appointment**: Represents a scheduled meeting between a patient and a physician.
  - id (UUID): Unique identifier for the appointment.
  - clinicId (Character Varying): Identifier for the clinic where the appointment takes place.
  - physicianId (FK - UUID): Foreign key linking to the Physician entity.
  - patientId (FK - UUID): Foreign key linking to the Patient entity.
  - startTime (Timestamp Without Time Zone): The start time of the appointment.
  - endTime (Timestamp Without Time Zone): The end time of the appointment.
  - status (Character Varying): Current status of the appointment (e.g., 'booked', 'cancelled').
- **AvailabilityBlock**: Defines periods when a physician is available.
  - id (UUID): Unique identifier for the availability block.
  - physicianId (FK - UUID): Foreign key linking to the Physician entity.
  - clinicId (Character Varying): Identifier for the clinic the block applies to.
  - startTime (Timestamp Without Time Zone): The start time of the availability block.
  - endTime (Timestamp Without Time Zone): The end time of the availability block.
  - type (Character Varying): Type of block (e.g., 'available', 'break', 'meeting').
- **BillingRule**: Stores rules related to billing, particularly for appointment gaps.
  - id (UUID): Unique identifier for the billing rule.
  - region (Character Varying): The geographical region to which the rule applies (e.g., 'Ontario').
  - gapDurationMinutes (Integer): The minimum required time gap between appointments in minutes.
  - beforeAppointmentBuffer (Integer): Additional buffer time required before an appointment in minutes.
  - afterAppointmentBuffer (Integer): Additional buffer time required after an appointment in minutes.

Relationships:

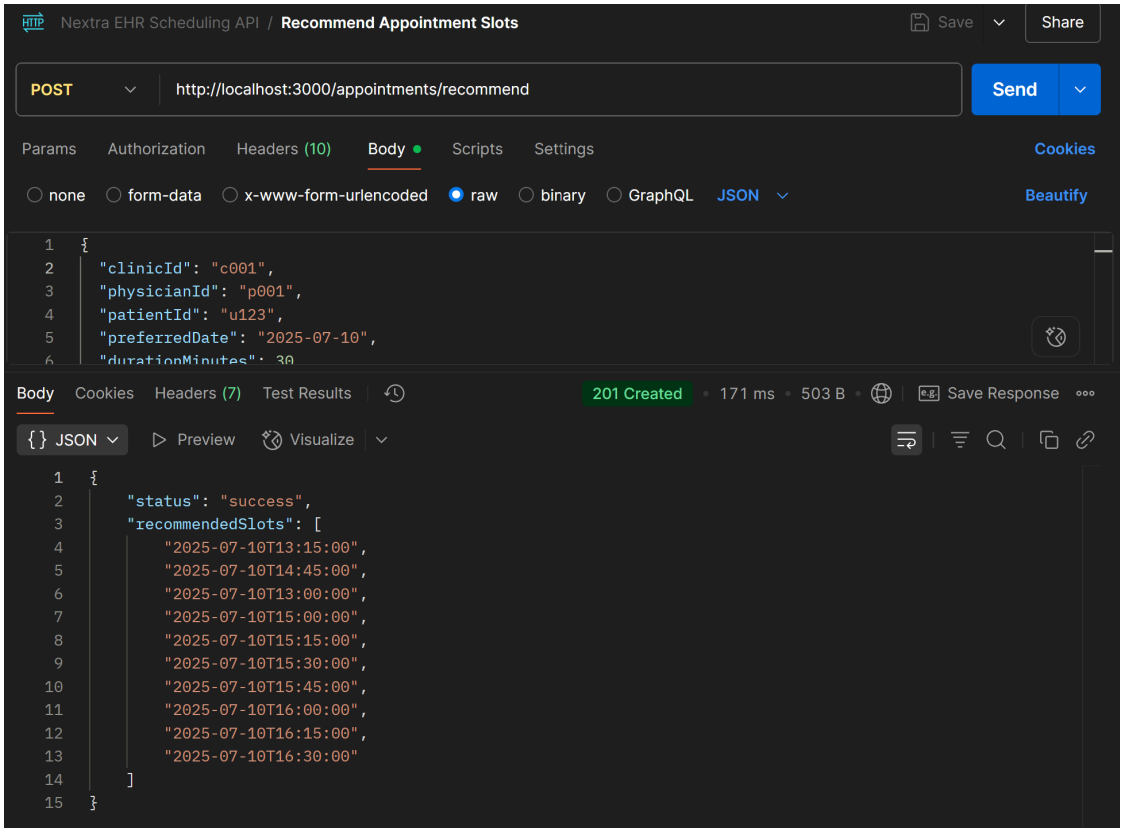
- A Physician can have multiple Appointments and multiple AvailabilityBlocks.
- A Patient can have multiple Appointments.
- An Appointment is linked to one Physician and one Patient, and occurs at a specific Clinic.
- An AvailabilityBlock is associated with one Physician and a Clinic.
- BillingRule entities are referenced by region to apply specific gap rules during appointment scheduling.

2. Key API

The primary API endpoint for appointment scheduling recommendations is detailed below.

POST /appointments/recommend

This endpoint is designed to suggest the top 10 available time slots for a specific patient and physician on a preferred date, considering various scheduling constraints.



3. Scheduling Algorithm: Detailed Logic

The scheduling algorithm is the core intelligence behind the appointment recommendation system. It aims to find the "least disruptive" slots by considering multiple factors and applying a scoring mechanism.

Step-by-Step Process:

1. **Input Validation & Data Fetching**:
    - The system first validates all incoming request parameters to ensure they are present and in the correct format.
    - It then fetches critical data: the specified physician's working hours for the preferred date, all AvailabilityBlock entries for that physician and clinic on the given date, and all existing Appointment records for the physician and clinic on the same date.
    - It also retrieves the relevant BillingRule for the specified region (e.g., Ontario) to apply specific gap requirements.
2. **Generate Candidate Slots**:
    - For each identified AvailabilityBlock, the algorithm iterates through the time period in increments equal to the minimum required gap (as defined by the BillingRule). This generates a series of potential start times for new appointments.
3. **Filter & Enforce Rules**:
    - Each generated candidate slot undergoes rigorous validation:
        - It must fit entirely within the physician's defined working hours.
        - It must not overlap with any existing Appointment.
        - It must adhere to the gapDurationMinutes, beforeAppointmentBuffer, and afterAppointmentBuffer specified in the BillingRule. Slots that are too close to other existing appointments (or the start/end of the day, considering buffers) are discarded to avoid clustering appointments too tightly.
4. **Score Valid Slots**:
    - For the remaining valid candidate slots, a scoring mechanism is applied to determine their "disruptiveness" (lower scores are better). The scoring factors include:
        - **Penalty for Proximity**: Slots immediately adjacent to existing appointments incur a penalty (e.g., -10 points).
        - **Penalty for Edge Slots**: Slots at the very beginning or end of the physician's working day receive a minor penalty (e.g., -5 points).
        - **Reward for Gap Filling**: Slots that effectively fill a gap between two existing appointments are rewarded (e.g., +10 points).
        - **Bonus for Large Gaps**: A bonus is added based on the size of the available gap before and after the slot, encouraging recommendations in less constrained periods (e.g., +Math.min(minPrevGap, minNextGap) / 60000).
    - Each valid slot is stored along with its calculated score.
5. **Select Top 10**:
    - Finally, all valid and scored slots are sorted in ascending order based on their score (lower score indicates a more optimal, less disruptive slot).
    - The algorithm then returns the top 10 recommended slots to the patient.

Example & Edge Cases:

- **Example Scenario**:
  - A physician has existing appointments from 09:30-10:00 and 11:00-11:30.
  - Their available block is 09:00-12:00.
  - A 30-minute slot from 10:15-10:45 would likely receive a better score and be recommended first because it is positioned furthest from both existing appointments, minimizing disruption.
- **Edge Case 1: Fully Booked Day**: If a physician has back-to-back appointments throughout their entire working day, the algorithm will correctly identify no available slots, and thus, no recommendations will be returned.
- **Edge Case 2: Insufficient Gap**: If a small gap exists, but it is less than the required appointment duration or fails to meet the minimum billing gap requirements, no slots will be returned for that period.
- **Edge Case 3: Multiple Equally Good Slots**: In scenarios where several slots have identical (or very similar) optimal scores, the system will return the first 10 slots based on their score and chronological order.

4. Assumptions

The design and implementation are based on the following key assumptions:

- Each physician is uniquely identified by a composite key consisting of their physicianId and the clinicId they are associated with.
- Billing rules are structured to be region-based and are designed for easy extensibility, allowing for the addition of new provincial or regional rules without significant code changes.
- The modular nature of the system ensures it is ready for future expansion, including the onboarding of more clinics, providers, and geographical regions, with minimal modifications required.
- Comprehensive seed data is supplied to facilitate demonstration, testing, and initial setup of the application.
- All times are handled in UTC to avoid timezone issues.
- Only availability blocks of type available are considered for slot recommendations.
- Billing gap and buffer rules are based on the Ontario region and are configurable.
- The slot search increments by the billing gap duration for efficiency.
- The scoring system penalizes clustering, edge-of-day slots, and squeezed slots, and prefers least disruptive options.

5. Integration & Unit Tests:

Comprehensive unit and integration tests were developed to ensure the reliability and correctness of the scheduling logic and API endpoints. The tests cover:

- Appointment slot recommendation logic, including edge cases for clustering, billing gaps, and least disruptive slot selection.
- Multi-clinic and multi-provider scenarios to verify correct scoping and data isolation.
- Controller endpoints, validating request handling and response formats.
- Seed data and business rules, ensuring the system behaves as expected with realistic data.
- All tests pass successfully, demonstrating robust coverage and confidence in the system's core functionality.