Doctor Appointment Booking API

Express + MongoDB (Mongoose) | Idempotency + ETag + Pagination + Logging

This project is intentionally kept simple for teaching, while still demonstrating four production-grade patterns that naturally appear in real systems:

- Idempotency: prevents duplicate bookings when clients retry after timeouts or double-clicks.
- ETag + If-Match/If-None-Match: enables caching (304) and optimistic concurrency control (412/428).
- Pagination: supports both offset pagination (page/limit) and cursor pagination (fast infinite scroll).
- Logging: request correlation + structured access logs, plus audit logs for “who did what and when”.

Project Overview

Domain: A clinic-style appointment system where patients pick a doctor and book an available time slot. The API creates doctors and pre-generates AVAILABLE slots via a seed route so demonstrations and tests are deterministic.

Data Model (MongoDB Collections)

The system stores data in these collections:

Key	Description
Doctor	Doctor profile (name, specialty, city, fee). Used for listing and showing ETag updates on profile edits.
Slot	A time slot for a doctor (startAt/endAt/status). Availability endpoint returns AVAILABLE slots for a date.
Appointment	Links patientId + doctorId + slotId with a status (REQUESTED/CONFIRMED/CANCELLED/PAYMENT_FAILED).
IdempotencyRecord	Stores booking replay data keyed by Idempotency-Key + scope and request body hash.
AuditLog	Business events (appointment.created/cancelled/rescheduled, payment.failed, sms.result, doctor.updated).

Setup and Run

Prerequisites

- Node.js 18+ (recommended)
- MongoDB running locally (or a reachable MongoDB URI)

Install and Start

```
npm install
npm start
```

Server runs at: http://localhost:3000

Environment Variables

Key	Description
PORT	HTTP port (default: 3000).
MONGODB_URI	MongoDB connection string (default: mongodb://127.0.0.1:27017/doctor_appointments).
ENABLE_TEST_ROUTES	Enable /test/reset and /test/seed (default: true).
SEED_BASE_DATE	Default YYYY-MM-DD for seeding if baseDate query param is not provided.

Health Check

```
GET http://localhost:3000/health
```

API Endpoints

Method	Path	Purpose	Patterns
GET	/doctors	List doctors (filter by city/specialty). Supports offset or cursor pagination.	Pagination
GET	/doctors/:id	Get a doctor with ETag; supports If-None-Match -> 304.	ETag (cache)
PUT	/doctors/:id	Update a doctor; requires If-Match; returns new ETag.	ETag (concurrency) + Audit
GET	/doctors/:id/availability	List AVAILABLE slots for a given date (YYYY-MM-DD). Supports offset or cursor.	Pagination
GET	/appointments	List appointments (filter by doctorId/patientId/status). Supports offset or cursor.	Pagination
GET	/appointments/:id	Get appointment with ETag; supports If-None-Match -> 304.	ETag (cache)
POST	/appointments	Book appointment.	Idempotency +

		Requires Idempotency-Key. Uses slot atomic booking + mock payment + mock SMS.	Audit
PATCH	/appointments/:id	Cancel or reschedule. Requires If-Match. Updates slots safely and writes audit logs.	ETag (concurrency) + Audit
GET	/audit/logs	List audit events (filter by entityType/entityId/action). Supports pagination.	Logging (audit) + Pagination
POST	/test/reset	Delete all collections (test/dev only).	Test utility
POST	/test/seed	Create 3 doctors and AVAILABLE slots for N days from baseDate.	Test utility

Thunder Client Demo Walkthrough (Recommended Class Flow)

Step 1 - Reset and Seed Deterministic Data

Use seed to create sample doctors and AVAILABLE slots. Slots exist BEFORE any appointment is booked; booking only changes a slot from AVAILABLE to BOOKED.

```
POST http://localhost:3000/test/reset

POST http://localhost:3000/test/seed?days=14&baseDate=2026-01-17
# Tip: If you want Jan 17 to appear in availability, baseDate must be
2026-01-17 (not 2026-17-01).
```

Step 2 - Get doctorId

```
GET http://localhost:3000/doctors?page=1&limit=10
```

Copy any doctor _id from items[]. This is your doctorId.

Step 3 - Get slotId (Availability)

Request AVAILABLE slots for a specific date in YYYY-MM-DD format. Choose any slot with status=AVAILABLE and copy its _id as slotId.

```
GET http://localhost:3000/doctors/<doctorId>/availability?date=2026-01-
17&limit=10
```

If you get an empty list, verify one of these:

- The date is inside the seeded range. Example: baseDate=2026-01-17 with days=14 seeds 2026-01-17 through 2026-01-30 (inclusive range depends on days, but Jan 17 is guaranteed).

- You used a valid date format (YYYY-MM-DD). “2026-17-01” is invalid (17 is not a month).
- You seeded after resetting. If you reseeded without using the latest doctorId, you might be querying a doctor that no longer exists.

Step 4 - Book Appointment (Idempotency)

Idempotency prevents duplicate bookings when clients retry due to timeouts. In Thunder Client, set Idempotency-Key header.

```
POST http://localhost:3000/appointments
Headers:
  Content-Type: application/json
  Idempotency-Key: booking-001
Body (JSON):
{
  "patientId": "p1",
  "doctorId": "<doctorId>",
  "slotId": "<slotId>",
  "notes": "First visit"
}
```

Retry the same request (same Idempotency-Key and body). The server returns the same response and adds header Idempotent-Replay: true.

Step 5 - ETag caching (If-None-Match -> 304)

```
GET http://localhost:3000/appointments/<appointmentId>
# Copy ETag response header

GET http://localhost:3000/appointments/<appointmentId>
Header: If-None-Match: "<etag>"
# If unchanged => 304 Not Modified
```

Step 6 - ETag optimistic concurrency (If-Match -> 412/428)

Updates require If-Match. Missing If-Match => 428. Stale If-Match => 412.

```
PATCH http://localhost:3000/appointments/<appointmentId>
Headers:
  Content-Type: application/json
  If-Match: "<latest-etag>"
Body (JSON):
{ "action": "cancel", "reason": "Not feeling well" }
```

Reschedule example:

```
PATCH http://localhost:3000/appointments/<appointmentId>
Headers: If-Match: "<latest-etag>"
Body (JSON):
{ "action": "reschedule", "newSlotId": "<newSlotId>" }
```

Step 7 - Pagination demos

Offset pagination is simple for “jump to page” UIs; cursor pagination is better for infinite scrolling and large datasets.

```
GET http://localhost:3000/doctors?page=1&limit=2
GET http://localhost:3000/doctors?page=2&limit=2

GET http://localhost:3000/doctors?mode=cursor&limit=2
# Use nextCursor from response
GET
http://localhost:3000/doctors?mode=cursor&limit=2&cursor=<nextCursor>
```

Step 8 - Audit logs (business logging)

```
GET http://localhost:3000/audit/logs?limit=10
```

Audit logs answer “who did what and when”. Example actions: appointment.created, appointment.cancelled, appointment.rescheduled, payment.failed, sms.result, doctor.updated.

Design Patterns Explained Using This Project

Idempotency (Safe Retries)

Where used: POST /appointments.

How it works in code:

- Client sends Idempotency-Key header.
- Server scopes it by (method + route + patientId) so keys do not collide across different patients.
- Server hashes the request body. Reusing the same key with a different body returns 409.
- If a previous request completed, server replays stored response (Idempotent-Replay: true).
- Records are kept for ~24 hours (TTL) for teaching purposes.

ETag (Caching + Optimistic Concurrency)

Where used: GET/PUT /doctors/:id and GET/PATCH /appointments/:id.

ETag format in this project:

```
"<prefix>-<mongoId>-<updatedAtMillis>"
```

Rules:

- If-None-Match equals current ETag => 304 Not Modified (client can reuse cached body).
- If-Match missing on update => 428 Precondition Required.
- If-Match stale (does not equal current ETag) => 412 Precondition Failed.

Pagination (Offset + Cursor)

Offset pagination: page + limit. Cursor pagination: a token points to the last item from the previous page.

Cursor token contains two fields to avoid duplicates:

- createdAt (or ts/startAt depending on list)
- id (MongoDB _id as a tie-breaker)

Logging (Access Logs + Audit Logs)

Access logs: one structured JSON log per request, including requestId for correlation.

Audit logs: domain events stored in MongoDB so you can query history for appointments and doctor changes.

Error Codes and What They Mean

Key	Description
304 Not Modified	Returned when If-None-Match matches current ETag (no response body).
400 Bad Request	Missing required parameters (e.g., missing date, missing patientId/doctorId/slotId).
402 Payment Required	Mock payment failed during booking (appointment marked PAYMENT_FAILED and slot freed).
404 Not Found	Doctor/Slot/Appointment not found.
409 Conflict	Slot not available, idempotency key reused with different body, or prior attempt failed.
412 Precondition Failed	ETag mismatch on update (resource changed since you fetched it).
428 Precondition Required	Missing If-Match header on update endpoints.
202 Accepted	Idempotent request still IN_PROGRESS (client should retry later).

Troubleshooting and Common Mistakes

Availability returns empty items

- Verify you seeded a range that includes the requested date.
- Use a valid date format: YYYY-MM-DD. Example: 2026-01-17 (not 2026-17-01).
- If you reseeded, fetch a fresh doctorId from GET /doctors (old IDs may no longer exist).
- If you still see 0 items, check MongoDB collection Slot to confirm documents exist for that doctorId and date.

ETag not visible in client

In Thunder Client, check the Response Headers tab. In browser-based clients, ETag is exposed via Access-Control-Expose-Headers in server.js.

Idempotency replay not happening

- Ensure the second request uses the same Idempotency-Key AND the same request body.
- If you change body with same key, server returns 409.
- If the first attempt failed, server returns 409 and you must use a new key to retry.