1 EV CHARGING STATION BOOKING SYSTEM — TECHNICAL STATUS REPORT (UPDATED)

System: IIS-hosted .NET 8 Web API (fat service design)
Database: MongoDB Atlas
Date: October 2025
Scope: Backend Web API implementation progress, logic, and workflows

---

1. Introduction & Scope

This backend powers the EV Charging Station Booking System, designed with a **fat service architecture** (all domain/business rules reside in the API, while clients are thin Web/Android UIs). The system now supports **role-aware, JWT-secured** operations across all actors:

- **System Admin** – manages platform, approves BackOffice applications.
- **BackOffice** – business operator (approved via application/review).
- **Station Operator** – staff assigned to specific stations.
- **EV Owner** – vehicle owner, books charging slots.

Security: JWT authentication with role claims, operator station scoping, and HMAC-secured QR tokens.
Core constraints: look-ahead booking horizon, cutoff windows, station schedules, approval workflows, QR issuance, and capacity enforcement.

---

2. Backend Architecture Overview

Project Layout

- **Controllers**
  AuthController, AdminController, BackOfficeController, StationController, BookingController,
  QrController, SessionsController, EvOwnerController, OperatorController, AuditsController, ReportsController, HealthController
- **Services**
  AuthService, AdminService, BackOfficeService, StationService, BookingService, SessionService, ScheduleService, QrTokenService, OperatorService, AuditService, NotificationService, InventoryService + InventoryRegenerator, PolicyService

- **Repositories**
  EvOwnerRepository, StationRepository, BookingRepository, AuditRepository
- **Infrastructure & Options**
  Mongo bootstrap, JwtTokenService, validators (NIC, Email, Phone, Geo, Schedule), mappings,
  options (JwtOptions, BookingOptions, PolicyOptions, InventoryOptions, CorsOptions).

Architectural Principles

- **Fat service**: Controllers are thin, services enforce business logic, repositories isolate persistence.
- **JWT claims**: role + operator station scope serve as the main policy boundary.
- **Inventory engine**: background service maintains slot availability.
- **Hosted workers**: InventoryRegenerator, NoShowSweeper.

---

3. Implemented Endpoints & Business Logic

A. Authentication

- **POST /api/Auth/login** – unified login (NIC for Owners; email for others).
- **Response**: JWT + claims (roles, operator station scope).

B. Admin

- **POST /api/Admin/admins** – create Admins.
- **GET /api/Admin/backoffice-applications** – list by status.
- **PUT /api/Admin/backoffice-applications/{nic}/approve|reject** – review flow.

C. BackOffice

- **POST /api/BackOffice/apply** – onboarding application.
- **POST /api/BackOffice/operators** – create operators (with station assignment).
- **PUT /api/BackOffice/operators/{nic}/stations** – update operator station scope.
- **GET /api/BackOffice/stations** – list owned stations.

D. Stations

- **POST /api/Station** – create (ownership stamped with BackOfficeNic).
- **PUT /api/Station/{id}/deactivate** – blocked if future bookings exist (Policy guard).

- **GET /api/Station/nearby** – 2dsphere search + 7-day availability summary.
- **PUT /api/Station/{id}/schedule** – weekly schedule with exceptions.

E. EV Owners

- **POST /api/EvOwner** – register by NIC.
- **PUT /api/EvOwner/{nic}/deactivate|reactivate** – lifecycle mgmt.

F. Bookings

- **POST /api/Booking** – creates booking if within horizon; auto-approve if enabled.
- **PUT /api/Booking/{id}** – reschedule (cutoff enforced).
- **DELETE /api/Booking/{id}** – cancel (cutoff enforced).
- **PUT /api/Booking/{id}/approve|reject** – BackOffice/Admin review.

G. QR

- **POST /api/Qr/issue/{bookingId}** – issue raw QR token (hash stored only).
- **POST /api/Qr/verify** – validate QR for operators.

H. Sessions

- **POST /api/Sessions/checkin** – QR validate, policy window, station scope.
- **POST /api/Sessions/finalize** – compute totals, return SessionReceipt.

I. Operator Ops

- **GET /api/Operator/inbox?date=** – daily Approved bookings.
- **POST /api/Operator/scan** – wrapper for check-in.
- **POST /api/Operator/exception** – mark NoShow, Aborted, CancelOnSite.

J. Audits & Notifications

- **POST /api/Audits/search** – list audit events by filters.
- Audit & notification emission on key transitions (Approve, Reject, Cancel, QrIssued, CheckedIn, Completed, NoShow).

K. Reports

- **GET /api/Reports/summary** – KPIs.
- **GET /api/Reports/timeseries/bookings|revenue** – trend charts.
- **GET /api/Reports/stations/{id}/utilization** – reserved vs capacity.

- **GET /api/Reports/occupancy-heatmap** – average reserved percentage by hour/dow.

---

4. Schemas & Data Models

Core DTOs

- **AuthLoginResponse**, **LoginRequest**, **OwnerRegisterRequest**, **BookingCreateRequest**, **BookingResponse**,
- **BookingApprovalResponse** (includes qrToken, qrExpiresAtUtc),
- **QrVerifyRequest / QrVerifyResponse**, **SessionCheckInRequest**, **SessionFinalizeRequest**, **SessionReceiptResponse**,
- **OperatorInboxItem**, **OperatorExceptionRequest**.

Entities

- **Owner**: Roles, IsActive, BackOfficeProfile (ApplicationStatus), OperatorStationIds.
- **Station**: BackOfficeNic, schedules, pricing, timezone, AutoApproveEnabled.
- **Booking**: OwnerNic, StationId, SlotStart/End, Status (Pending, Approved, CheckedIn, Completed, Cancelled, Rejected, NoShow, Aborted), QrTokenHash, QrExpiresAtUtc.
- **Session**: bookingId, stationId, checkInUtc, completedAtUtc, energyKwh, unitPrice, total.
- **Audit**: entityType, entityId, action, actorNic, actorRole, payload, createdAtUtc.
- **Notification**: type, toNic, subject, message, payload.

---

5. Key Business Modules

5.1 Slot Inventory (Background Engine)

- **InventoryService** + **InventoryRegenerator** maintain rolling horizon slots (default 14 days).
- Atomic reserve/release on booking create/update/cancel.
- Unique compound index (StationId, SlotStartUtc) ensures no double-booking.

5.2 Idempotency

- **Idempotency-Key** header (persisted with TTL 7 days).
- Same key → same response returned.
- Applied to booking creation (and extendable to sessions).

5.3 No-Show Automation

- **NoShowSweeper** hosted service runs periodically.
- Marks Approved bookings as NoShow after slot end + 15 min grace.
- Releases inventory, emits audits/notifications.

5.4 Audits & Notifications

- **AuditService** logs all lifecycle transitions.
- **NotificationService** queues stubs (Email/SMS/Push adapters pluggable).
- Searchable via /api/Audits/search.

5.5 Policy Guardrails

- **PolicyService** enforces:
    - Booking horizon (≤ 7 days)
    - Modify/Cancel cutoff (12h default)
    - EarliestCheckIn (15m before start)
    - Station deactivation guard if future bookings exist
- Violations → 409 Conflict with machine-readable policy codes.

---

6. Security & Operational Hardening

- **JWT** with role + station scope claims.
- **QR tokens**: HMAC-based, raw never stored (SHA-256 only).
- **Input validation**: NIC, Email, Phone, Schedule.
- **CORS & HTTPS** configured via options/IIS.
- **Audit trail**: every critical action logged.
- **Logs & Observability**: structured logs, request IDs, metrics (planned).

---

7. Reporting & Dashboards

- **KPIs**: approvals, check-in rates, completion rates, revenue, energy.
- **Time series**: bookings/revenue by day/week/month.

- **Utilization**: reserved vs capacity (station-local).
- **Occupancy heatmap**: hourly % reserved by day of week.
- All restricted to Admin/BackOffice roles.

---

8. Current Status vs Guide

- ✅ Unified login & role scoping.
- ✅ BackOffice application workflow.
- ✅ Operator station scoping.
- ✅ Station CRUD + schedules + discovery.
- ✅ Booking lifecycle + QR issuance.
- ✅ Sessions check-in/finalize.
- ✅ Slot inventory engine.
- ✅ No-Show sweeper.
- ✅ Audits & notifications.
- ✅ Policy guardrails.
- ✅ Reporting & dashboards.