technical manual full

Signox LogX Team

2025-07-09

Table of Contents

# 1 Smart Roster System - Comprehensive Technical Manual

This document consolidates all technical manuals for the Smart Roster backend modules, providing a single, in-depth reference for developers, system integrators, and technical administrators.

## 1.1 Table of Contents

1. Attendance Module
2. Course Management Module
3. School Management Module
4. Timetable Management Module
5. User Management Module
6. Notifications Module
7. Reports & Analytics Module

## 1.2 1. Attendance Module

# 2 Attendance Module - Technical Manual

## 2.1 Overview

The Attendance module is responsible for all aspects of session attendance management, including session lifecycle, trainee check-in, facial recognition, geolocation validation, feedback, and analytics. It is a core backend component of the Smart Roster system.

## 2.2 Architecture

* **Models:**
  + AttendanceSession: Represents a scheduled or active session, linked to a timetable, trainer, class group, and room. Tracks session status, adherence, feedback, and makeup sessions.
  + AttendanceRecord: Represents a trainee’s attendance for a session, including status, geolocation, facial image, and feedback.
* **Services:**
  + AttendanceSessionService: Handles session lifecycle (start, end), makeup slot finding, trainer changes, and notifications. Implements business rules and constraints (timing, geolocation, facial recognition).
  + AttendanceRecordService: Handles trainee check-in, feedback, and status updates.
* **Views:**
  + REST API endpoints for session and record management, analytics, and feedback. Implements role-based access control.
* **Serializers:**
  + Multiple serializers for different use cases (basic, detail, create, feedback).

## 2.3 Key Features & Constraints

* **Session Lifecycle:**
  + Sessions move through scheduled → active → completed/canceled states.
  + Only assigned trainers can start/end sessions, with geolocation and facial recognition validation.
  + Strict timing windows for starting/ending sessions (e.g., 15 min before, 30 min after start).
* **Attendance Validation:**
  + Trainees check in with facial image and geolocation; system validates presence and identity.
  + Statuses: present, late, absent, excused, pending.
  + Absence can be excused with documentation and staff verification.
* **Feedback & Analytics:**
  + Trainees provide session feedback (content, delivery, engagement, overall rating).
  + Aggregated feedback metrics are stored on the session.
  + Analytics endpoints for attendance and feedback statistics.
* **Makeup Sessions:**
  + Missed/canceled sessions can be rescheduled as makeup sessions.
  + System finds available slots, avoiding conflicts, breaks, and room clashes.
* **Performance:**
  + Query optimizations with select\_related, prefetch\_related, and caching.
  + Batch operations for notifications and record updates.
* **Security:**
  + Role-based access enforced at API and query level.
  + All sensitive operations require authentication.

## 2.4 Integration Points

* **Timetable Management:** Links to scheduled sessions and room assignments.
* **User Management:** Trainer and trainee roles, facial encodings.
* **Notifications:** Sends notifications on session events.
* **Reporting:** Feeds analytics and reporting modules.

## 2.5 Data Flow

1. **Session Creation:** Triggered from timetable or manually for makeup.
2. **Session Start:** Trainer starts session (geolocation/facial validation), status changes to active.
3. **Trainee Check-in:** Trainee submits image/location, system validates and updates record.
4. **Session End:** Trainer ends session, system marks absentees, aggregates feedback.
5. **Feedback:** Trainees submit feedback, metrics update on session.

## 2.6 Error Handling & Logging

* All critical operations are wrapped in transactions.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 2.7 2. Course Management Module

# 3 Course Management Module - Technical Manual

## 3.1 Overview

The Course Management module handles the definition, structure, and administration of all courses, topics, subtopics, and their mapping to programmes, terms, and class groups. It is central to curriculum management and academic planning.

## 3.2 Architecture

* **Models:**
  + Course: Defines a course, its code, department, description, prerequisites, session structure, and links to programmes.
  + Topic & Subtopic: Hierarchical structure for course content, with ordering and duration.
  + ProgrammeCourseMapping: Maps courses to programmes, terms, years, and room types (theory/practical).
  + Enrollment: Connects class groups to courses for a specific academic year and term, assigns trainers, and tracks status.
* **Views:**
  + REST API endpoints for CRUD operations on courses, topics, mappings, and enrollments.
  + Custom endpoints for prerequisite trees, attendance statistics, and mass registration templates.
* **Serializers:**
  + Multiple serializers for different use cases (basic, detail, create, update).

## 3.3 Key Features & Constraints

* **Course Structure:**
  + Courses can have prerequisites (with circular dependency checks).
  + Each course can have multiple topics and subtopics, with enforced ordering and duration validation.
  + Courses are mapped to programmes, terms, and years, supporting flexible curriculum design.
* **Enrollment Management:**
  + Enrollments link class groups to courses, assign trainers, and track status (scheduled, in progress, completed, cancelled).
  + Validation ensures only valid curriculum mappings and trainer availability.
* **Room & Resource Mapping:**
  + Each course-programme mapping specifies required room types for theory and practical sessions.
* **Attendance Analytics:**
  + Attendance rates are calculated per course and enrollment, integrating with the attendance module.
* **Bulk Operations:**
  + Supports mass course registration via Excel templates and bulk upload endpoints.
* **Performance:**
  + Query optimizations with select\_related, prefetch\_related, and indexed fields.
* **Security:**
  + Role-based access enforced at API and query level.
  + All sensitive operations require authentication.

## 3.4 Integration Points

* **School Management:** Links to departments, programmes, class groups, and academic years.
* **User Management:** Assigns trainers and tracks trainee enrollments.
* **Attendance Module:** Feeds course and enrollment data for attendance tracking and analytics.
* **Timetable Management:** Used for scheduling sessions based on enrollments and mappings.

## 3.5 Data Flow

1. **Course Definition:** Admins define courses, topics, and subtopics.
2. **Curriculum Mapping:** Courses are mapped to programmes, terms, and years.
3. **Enrollment:** Class groups are enrolled in courses for a term/year, trainers are assigned.
4. **Session Scheduling:** Timetable and attendance modules use enrollments to schedule sessions.
5. **Analytics:** Attendance and performance data are aggregated per course and enrollment.

## 3.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 3.7 3. School Management Module

# 4 School Management Module - Technical Manual

## 4.1 Overview

The School Management module defines and manages the academic structure of the institution, including schools, departments, programmes, class groups, academic years, and terms. It is foundational for all academic and administrative operations.

## 4.2 Architecture

* **Models:**
  + School: Represents the institution, with a single instance enforced. Tracks departments, logo, and key statistics.
  + Department: Academic departments under a school, each with a head (HOD) and trainers.
  + Programme: Academic programmes under departments, with code, level, and class groups.
  + ClassGroup: Represents a cohort of trainees for a specific year and programme.
  + AcademicYear: Academic years (e.g., 2024/2025), with active status and linked terms.
  + Term: Academic terms (e.g., Term 1, Term 2), with start/end dates and validation for overlaps.
* **Views:**
  + REST API endpoints for CRUD operations on all models.
  + Custom endpoints for retrieving related data (e.g., terms for an academic year).
* **Serializers:**
  + Serializers for each model, supporting detail and list views.

## 4.3 Key Features & Constraints

* **Single School Instance:**
  + Only one school can exist in the system; deletion is prevented.
* **Department & Programme Structure:**
  + Departments are linked to a school; each has a HOD and trainers.
  + Programmes are unique per department and can have multiple class groups.
* **Class Groups:**
  + Each class group is linked to a programme and year, with a unique name and list of trainees.
* **Academic Calendar:**
  + Academic years and terms are strictly validated for date overlaps and active status.
  + Terms are linked to academic years and have duration and current status properties.
* **Statistics & Aggregation:**
  + Properties for total departments, programmes, class groups, trainees, and trainers at each level.
* **Performance:**
  + Query optimizations with aggregation and indexed fields.
* **Security:**
  + Role-based access enforced at API and query level.
  + All sensitive operations require authentication.

## 4.4 Integration Points

* **Course Management:** Departments and programmes are used for course mapping and enrollment.
* **User Management:** Assigns HODs, trainers, and trainees to departments, programmes, and class groups.
* **Timetable & Attendance:** Academic years, terms, and class groups are used for scheduling and attendance tracking.

## 4.5 Data Flow

1. **School/Department Setup:** Admin creates the school, departments, and assigns HODs/trainers.
2. **Programme & Class Group Creation:** Programmes are defined under departments; class groups are created for each cohort.
3. **Academic Calendar:** Academic years and terms are created and managed.
4. **Integration:** Other modules use this structure for course mapping, enrollment, and scheduling.

## 4.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 4.7 4. Timetable Management Module

# 5 Timetable Management Module - Technical Manual

## 5.1 Overview

The Timetable Management module is responsible for scheduling, managing, and validating all class sessions, room bookings, and trainer availabilities. It ensures conflict-free, optimized timetables for all class groups, trainers, and rooms.

## 5.2 Architecture

* **Models:**
  + Room: Defines physical/virtual teaching spaces, with capacity, facilities, geolocation, maintenance, and availability.
  + TrainerAvailability: Tracks when trainers are available for scheduling, with term-specific options and conflict checks.
  + Timetable: Represents scheduled sessions, linking enrollments to days, times, rooms, and session types. Handles recurring, draft, and makeup sessions.
  + ClassGroupSchedule: Cached weekly schedule for each class group and term, for fast access.
* **Views:**
  + REST API endpoints for CRUD operations on rooms, availabilities, timetables, and schedules.
  + Endpoints for conflict checking, alternative suggestions, and analytics.
* **Serializers:**
  + Serializers for each model, supporting detail, list, and update operations.

## 5.3 Key Features & Constraints

* **Room Management:**
  + Rooms have types, capacity, facilities, geolocation zones, maintenance schedules, and booking limits.
  + Geolocation validation for attendance and scheduling.
* **Trainer Availability:**
  + Trainers define available slots, with validation for overlaps and term specificity.
* **Timetable Scheduling:**
  + Timetable entries link enrollments to days, times, rooms, and session types (theory/practical).
  + Supports recurring, draft, and temporary (makeup) sessions.
  + Conflict detection for trainers, class groups, and rooms, with alternative suggestions.
* **Validation & Constraints:**
  + Strict validation for overlapping sessions, room bookings, and trainer/class group assignments.
  + Maintenance and geolocation constraints enforced for room usage.
* **Performance:**
  + Query optimizations, indexed fields, and schedule caching for fast access.
* **Security:**
  + Role-based access enforced at API and query level.
  + All sensitive operations require authentication.

## 5.4 Integration Points

* **Course Management:** Uses enrollments for session scheduling.
* **School Management:** Links to departments, class groups, and terms.
* **Attendance Module:** Provides session data for attendance tracking.
* **User Management:** Assigns trainers to sessions and availabilities.

## 5.5 Data Flow

1. **Room & Availability Setup:** Admins define rooms and trainer availabilities.
2. **Timetable Creation:** Sessions are scheduled, validated for conflicts, and published.
3. **Conflict Resolution:** System checks for conflicts and suggests alternatives.
4. **Schedule Access:** Class groups, trainers, and rooms access their schedules via API or UI.
5. **Integration:** Timetable data is used for attendance, reporting, and analytics.

## 5.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 5.7 5. User Management Module

# 6 User Management Module - Technical Manual

## 6.1 Overview

The User Management module handles all aspects of user identity, authentication, roles, and profiles. It supports role-based access, facial recognition, and profile management for trainees, trainers, HODs, and administrators.

## 6.2 Architecture

* **Models:**
  + CustomUser: Extends Django’s user model with email login, roles, gender, phone, ID, address, and facial recognition fields.
  + TraineeProfile, TrainerProfile, HODProfile, DPAcademicsProfile: One-to-one profiles for each user role, with role-specific fields (e.g., trainee status, teaching load, department assignment).
* **Views:**
  + REST API endpoints for user CRUD, registration, profile management, and bulk operations.
  + Endpoints for facial image upload and verification.
* **Serializers:**
  + Serializers for user creation, update, detail, and each profile type.

## 6.3 Key Features & Constraints

* **Role-Based Access:**
  + Users have roles: trainee, trainer, HOD, DP Academics, admin, etc.
  + Role-specific profile models and permissions.
* **Authentication & Registration:**
  + Email-based login, password validation, and email verification.
  + Bulk registration via Excel upload, with email notifications.
* **Facial Recognition:**
  + Users can register a facial image; encoding is stored for attendance and security.
  + Dlib-based face encoding and validation (single face per image enforced).
* **Profile Management:**
  + Each user has a detailed profile with academic, professional, and contact info.
  + Trainee and trainer profiles track status, qualifications, and teaching load.
* **Security:**
  + Password validation, email verification, and role-based permissions.
  + All sensitive operations require authentication.
* **Performance:**
  + Query optimizations and bulk operations for mass registration.

## 6.4 Integration Points

* **School Management:** Assigns users to departments, programmes, and class groups.
* **Attendance Module:** Uses face encodings for attendance validation.
* **Course Management:** Assigns trainers to courses and enrollments.

## 6.5 Data Flow

1. **User Registration:** Admins or users register via API or bulk upload.
2. **Profile Completion:** Users complete their profile and upload a facial image.
3. **Role Assignment:** Admins assign roles and link users to departments, class groups, etc.
4. **Authentication:** Users log in with email and password; email verification required.
5. **Integration:** User data is used by other modules for scheduling, attendance, and analytics.

## 6.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 6.7 6. Notifications Module

# 7 Notifications Module - Technical Manual

## 7.1 Overview

The Notifications module manages all user and system notifications, including in-app, email, and push notifications. It supports individual, broadcast, and preference-based notifications for all user roles.

## 7.2 Architecture

* **Models:**
  + Notification: Stores individual notifications for users, with type, priority, status, and related object references.
  + NotificationPreference: User preferences for notification types and delivery channels (email, push, in-app).
  + BroadcastNotification: Notifications sent to multiple users or groups (roles, departments, class groups, etc.).
* **Views:**
  + REST API endpoints for listing, marking as read/unread, and managing notification preferences.
  + Endpoints for sending and broadcasting notifications.
* **Serializers:**
  + Serializers for notification creation, update, and preference management.

## 7.3 Key Features & Constraints

* **Notification Types:**
  + Attendance, report, account, session, and system notifications.
  + Priority levels: low, medium, high.
* **Delivery Channels:**
  + In-app, email, and push notifications, based on user preferences.
* **Broadcasting:**
  + Send notifications to all users, by role, department, school, or class group.
* **Status Tracking:**
  + Mark notifications as read/unread, deleted, with timestamps.
* **Integration:**
  + Linked to any object in the system via generic foreign keys.
* **Performance:**
  + Indexed fields for fast retrieval and filtering.
* **Security:**
  + Only authorized users can send or broadcast notifications.

## 7.4 Integration Points

* **Attendance, Reports, Account, and Session modules:** Trigger notifications for key events (e.g., session start, report ready, account changes).
* **User Management:** Uses user roles and preferences for targeting and delivery.

## 7.5 Data Flow

1. **Event Trigger:** System or user action triggers a notification.
2. **Notification Creation:** Notification is created and delivered via enabled channels.
3. **User Interaction:** User reads, marks as read/unread, or deletes notification.
4. **Preference Management:** Users set preferences for notification types and channels.

## 7.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.

## 7.7 7. Reports & Analytics Module

# 8 Reports & Analytics Module - Technical Manual

## 8.1 Overview

The Reports & Analytics module generates, stores, and delivers attendance, performance, and operational reports at all levels (course, trainee, trainer, class group, programme, department, school). It supports both scheduled and on-demand reporting, with advanced analytics and export options.

## 8.2 Architecture

* **Models:**
  + BaseReport: Abstract base for all report types, with common metrics and metadata.
  + CourseReport, TraineeReport, TrainerReport, ClassGroupReport, ProgrammeReport, DepartmentReport, SchoolReport: Specialized reports for each entity, with attendance, punctuality, and performance metrics.
* **Views:**
  + REST API endpoints for generating, retrieving, and exporting reports.
  + Endpoints for analytics dashboards and scheduled report delivery.
* **Serializers:**
  + Serializers for each report type, supporting detail, list, and export formats.

## 8.3 Key Features & Constraints

* **Comprehensive Reporting:**
  + Reports at all levels: course, trainee, trainer, class group, programme, department, school.
  + Metrics: attendance %, punctuality, session counts, average durations, and more.
* **Advanced Analytics:**
  + Aggregated and comparative analytics across time periods and entities.
  + JSON data fields for flexible analytics and dashboard integration.
* **Scheduling & Automation:**
  + Reports can be generated automatically (daily, weekly, monthly) or on demand.
* **Export & Integration:**
  + Reports exportable as PDF, Excel, CSV, or JSON.
  + Integration with notification system for report delivery.
* **Performance:**
  + Indexed fields for fast retrieval and filtering.
* **Security:**
  + Role-based access enforced at API and query level.
  + All sensitive operations require authentication.

## 8.4 Integration Points

* **Attendance, Course, School, and User modules:** Source data for all reports and analytics.
* **Notifications:** Sends alerts when new reports are available.

## 8.5 Data Flow

1. **Data Collection:** Attendance and performance data is collected from all modules.
2. **Report Generation:** Reports are generated automatically or on demand.
3. **User Access:** Users access reports via dashboards or receive them via notifications.
4. **Export & Analysis:** Reports can be exported or integrated into analytics dashboards.

## 8.6 Error Handling & Logging

* All critical operations are validated before saving.
* Errors are logged with context for debugging.
* User-facing errors are descriptive and actionable.