

# School Management System — Administrator Manual

## School Management System — Administrator Manual

Version: 1.0 Last Updated: March 2026

---

### Table of Contents

- [1. Introduction](#)
  - [2. System Architecture](#)
  - [3. Installation & Deployment](#)
  - [4. Initial Setup & Configuration](#)
  - [5. User Management](#)
  - [6. Role & Permission Management](#)
  - [7. Multi-Tenancy Administration](#)
  - [8. Academic Configuration](#)
  - [9. Billing & Fee Structure Setup](#)
  - [10. Settings & Lookup Data](#)
  - [11. Reports Administration](#)
  - [12. Database Management](#)
  - [13. Background Services](#)
  - [14. Security & Authentication](#)
  - [15. Troubleshooting](#)
- 

## 1. Introduction

This manual is intended for **system administrators** responsible for deploying, configuring, and maintaining the School Management System (SMS). It covers infrastructure setup, user and role management, multi-tenancy, billing configuration, and ongoing maintenance.

### 1.1 Audience

- IT administrators managing the SMS deployment
- Super administrators managing multiple tenant schools
- School administrators configuring their tenant

### 1.2 System Requirements

| Component | Requirement                                           |
|-----------|-------------------------------------------------------|
| Runtime   | .NET 10.0 or later                                    |
| Database  | SQL Server 2019+ or Azure SQL                         |
| OS        | Windows Server 2019+, Linux (Ubuntu 22.04+), or macOS |
| Memory    | 2 GB RAM minimum, 4 GB recommended                    |
| Browser   | Chrome 90+, Firefox 90+, Edge 90+, Safari 15+         |

---

## 2. System Architecture

### 2.1 Solution Projects

The SMS solution consists of three projects:

| Project            | Purpose                                                    | Ports                  |
|--------------------|------------------------------------------------------------|------------------------|
| <b>SMSAPP.Data</b> | Data layer — EF Core models, migrations, seeders, services | N/A (library)          |
| <b>SMSAPP.Mvc</b>  | Web application — controllers, views, authentication       | HTTP 5125 / HTTPS 7218 |
| <b>SMSAPP.Api</b>  | REST API — Swagger documentation, external integrations    | HTTP 5228 / HTTPS 7220 |

## 2.2 Key Architectural Patterns

- **Cookie-based authentication** with 8-hour sliding expiration
- **Permission-based authorization** using claims and policies
- **Multi-tenancy** with row-level data isolation via TenantId
- **Tenant resolution** via subdomain (e.g., schoolname.smsapp.com)
- **Background services** for async operations (e.g., student registration migration)
- **QuestPDF** for report generation with tenant-branded logos

## 2.3 Data Layer

- **SQL Server** with Entity Framework Core 10
- Default schema: dbo
- All entities use Guid primary keys with newid() defaults
- Auto-migration on application startup
- Partial ApplicationDbContext classes split across multiple files

# 3. Installation & Deployment

## 3.1 Prerequisites

```
# Install .NET 10 SDK
# https://dotnet.microsoft.com/download/dotnet/10.0

# Verify installation
dotnet --version

# Restore tools (EF Core CLI)
dotnet tool restore
```

## 3.2 Building the Solution

```
# Build entire solution
dotnet build

# Build in Release mode
dotnet build -c Release

# Publish for deployment
dotnet publish src/SMSAPP.Mvc -c Release -o ./publish
```

## 3.3 Configuration Files

Configuration is managed via appsettings.json and environment-specific overrides.

**appsettings.json** — Base configuration:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=<host>;Database=<dbname>;User ID=<user>;Password=<passw"
  }
}
```

**appsettings.Development.json** — Development overrides including bootstrap admin credentials:

```
{
  "BootstrapAdmin": {
    "Username": "dev-admin",
    "Password": "DevAdmin123!"
  }
}
```

**Security:** Never deploy with development credentials. Always change the bootstrap admin password in production and use secure connection strings with proper encryption.

### 3.4 Running the Application

```
# Development — MVC with hot reload
dotnet watch --project src/SMSAPP.Mvc
```

```
# Development — API with hot reload
dotnet watch --project src/SMSAPP.Api
```

```
# Production
dotnet run --project src/SMSAPP.Mvc -c Release
dotnet run --project src/SMSAPP.Api -c Release
```

### 3.5 Deployment to IIS

1. Publish the application: `dotnet publish src/SMSAPP.Mvc -c Release -o ./publish`
2. Configure IIS with the ASP.NET Core Module
3. Point the site to the publish/ folder
4. Ensure the application pool is set to **No Managed Code**
5. Configure HTTPS bindings with a valid SSL certificate

### 3.6 Deployment to Linux

```
# Install .NET 10 runtime on Ubuntu
sudo apt-get update && sudo apt-get install -y dotnet-runtime-10.0
```

```
# Run as a systemd service
sudo nano /etc/systemd/system/smsapp.service
```

Example systemd unit file:

```
[Unit]
Description=SMS App MVC
After=network.target

[Service]
WorkingDirectory=/var/www/smsapp
ExecStart=/usr/bin/dotnet SMSAPP.Mvc.dll
Restart=always
RestartSec=10
SyslogIdentifier=smsapp
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

## 4. Initial Setup & Configuration

### 4.1 Database Initialization

The database is automatically initialized on first startup. The initialization sequence is:

| Order | Seeder                 | Purpose                                                                                              |
|-------|------------------------|------------------------------------------------------------------------------------------------------|
| 1     | AdmissionLookupSeeder  | Seeds lookup tables: nationalities, religions, relationships, occupations, qualifications, positions |
| 2     | BootstrapAdminSeeder   | Creates the initial admin user from appsettings config                                               |
| 3     | SmsAppSampleDataSeeder | Seeds sample data (development environment only)                                                     |

### 4.2 Bootstrap Admin Account

On first startup, the system creates an admin user from configuration:

- **Username:** As configured in `BootstrapAdmin:Username`
- **Password:** As configured in `BootstrapAdmin:Password`
- **Role:** Full system administrator with all permissions

**Important:** The bootstrap admin is forced to change their password on first login via the `MustChangePassword` claim.

## 4.3 First Login Checklist

After deploying and logging in for the first time:

1. **Change bootstrap admin password** — You will be redirected automatically
  2. **Create roles** — Define permission groups (see Section 6)
  3. **Create user accounts** — Add staff users with appropriate roles (see Section 5)
  4. **Configure academic structure** — Terms, classes, courses, subjects (see Section 8)
  5. **Set up fee structure** — Fee components, fee groups, class fee groups (see Section 9)
  6. **Configure settings** — Schools, holidays, incident types (see Section 10)
- 

## 5. User Management

**Path:** Security > Users **Required Permission:** `users.view (view)`, `users.manage (create/edit/reset/unlock)`

### 5.1 Creating a User

1. Navigate to **Security > Users**.
2. Click **+ New User**.
3. Fill in the required fields:
  - **Username** — Unique login identifier
  - **Full Name** — Display name
  - **Email** — Contact email
  - **Password** — Initial password (user will be forced to change on first login)
  - **Role(s)** — Assign one or more roles
4. Click **Create**.

### 5.2 Editing a User

1. Click the **Edit** icon on any user row.
2. Modify the user's profile, role assignments, or status.
3. Click **Save**.

### 5.3 Resetting a Password

1. Navigate to the user's profile.
2. Click **Reset Password**.
3. Enter the new temporary password.
4. Click **Reset**. The user will be forced to change it on next login.

### 5.4 Unlocking a Locked Account

If a user is locked out (e.g., too many failed login attempts):

1. Navigate to the user's profile.
2. Click **Unlock Account**.
3. The user can now log in again.

### 5.5 User Status

- **Active** — User can log in and access the system
  - **Inactive** — User account is disabled; login is denied
  - **Locked** — User is temporarily locked out; admin can unlock
- 

## 6. Role & Permission Management

### 6.1 Permission Categories

The system uses a granular permission model organized by functional area:

| Category             | View Permission    | Manage Permission               |
|----------------------|--------------------|---------------------------------|
| <b>Users</b>         | users.view         | users.manage                    |
| <b>Roles</b>         | roles.view         | roles.manage                    |
| <b>Permissions</b>   | permissions.view   | permissions.manage              |
| <b>Students</b>      | students.view      | students.manage                 |
| <b>Academics</b>     | academics.view     | academics.manage                |
| <b>Staff</b>         | staff.view         | staff.manage                    |
| <b>Billing</b>       | billing.view       | billing.manage                  |
| <b>Reports</b>       | reports.view       | reports.export                  |
| <b>Communication</b> | communication.view | communication.manage            |
| <b>Settings</b>      | settings.view      | settings.manage                 |
| <b>Tenants</b>       | tenants.view       | tenants.manage / tenants.create |

## 6.2 Viewing All Permissions

**Path:** Security > Permissions

This is a read-only catalog of all permissions in the system. Use it as a reference when building roles.

## 6.3 Creating a Role

**Path:** Security > Roles

1. Click **+ New Role**.
2. Enter a **Role Name** (e.g., "Teacher", "Accountant", "Head Teacher").
3. Check the permissions to assign:
  - **View permissions** allow read-only access to a module
  - **Manage permissions** allow create, edit, and delete operations
4. Click **Create**.

## 6.4 Recommended Role Templates

| Role                 | Permissions                                                                          |
|----------------------|--------------------------------------------------------------------------------------|
| <b>Super Admin</b>   | All permissions (auto-assigned)                                                      |
| <b>School Admin</b>  | Users, Roles, Students, Academics, Staff, Billing, Reports, Settings, Communication  |
| <b>Head Teacher</b>  | Students (view/manage), Academics (view/manage), Staff (view), Reports (view/export) |
| <b>Teacher</b>       | Students (view), Academics (view/manage), Reports (view)                             |
| <b>Accountant</b>    | Billing (view/manage), Reports (view/export), Students (view)                        |
| <b>Registrar</b>     | Students (view/manage), Academics (view)                                             |
| <b>Report Viewer</b> | Reports (view/export)                                                                |

## 6.5 Editing a Role

1. Navigate to the role's detail page.
2. Click **Edit**.
3. Modify the assigned permissions.
4. Click **Save**. Changes take effect on the user's next login.

**Note:** Removing a permission from a role immediately affects all users assigned to that role on their next page load.

# 7. Multi-Tenancy Administration

**Path:** Tenant Management **Required:** Super Admin claim

## 7.1 How Multi-Tenancy Works

The SMS supports a multi-tenant architecture where:

- Each school (tenant) has a **unique subdomain** (e.g., `riverview.smsapp.com`)
- Data is **isolated** at the row level using TenantId on every entity
- Tenants share the same database but cannot see each other's data
- **Super administrators** can view and manage all tenants

## 7.2 Tenant Resolution

The system resolves the current tenant through:

1. **Subdomain detection** — Extracts the subdomain from the request URL
2. **User claims** — Authenticated users carry their TenantId in their claims
3. **Super Admin override** — Super admins can optionally filter to a specific tenant

### 7.3 Creating a New Tenant

1. Navigate to **Tenant Management**.
2. Click **+ Create Tenant**.
3. Fill in:
  - **Name** — School name (e.g., “Riverview Academy”)
  - **Subdomain** — Unique subdomain prefix (e.g., riverview)
  - **Logo URL** — Path to the school’s logo file (relative to wwwroot/)
  - **Contact Email** — School admin email
  - **Contact Phone** — School phone number
  - **Address** — School physical address
  - **Currency** — Currency for billing (e.g., GHS)
4. Click **Create**.

### 7.4 Tenant Logo for Reports

Each tenant’s logo appears on exported PDF reports (upper-right corner):

- Set the LogoUrl property to a file path relative to wwwroot/ (e.g., uploads/tenants/riverview-logo.png)
- If no logo is set, the system falls back to wwwroot/branding/logo.png
- Supported formats: PNG, JPEG

### 7.5 Enabling / Disabling a Tenant

1. Navigate to the tenant’s detail page.
2. Click **Toggle Status**.
3. Disabled tenants cannot access the system — all users under that tenant are locked out.

### 7.6 Editing a Tenant

1. Navigate to the tenant’s detail page.
2. Click **Edit**.
3. Modify the tenant properties.
4. Click **Save**.

---

## 8. Academic Configuration

This section covers the building blocks that must be configured before the school can operate.

### 8.1 Terms

**Path:** Academic > Terms

Terms define the academic calendar. Create at least one term before registering students.

| Field             | Description                                                   |
|-------------------|---------------------------------------------------------------|
| <b>Name</b>       | Term name (e.g., “Term 1 2025/2026”)                          |
| <b>Start Date</b> | When the term begins                                          |
| <b>End Date</b>   | When the term ends                                            |
| <b>Status</b>     | Active or Inactive — Only one term should be active at a time |

### 8.2 Classes

**Path:** Academic > Classes

Classes represent the school’s organizational units (e.g., “Class 1A”, “JHS 2B”).

| Field            | Description                   |
|------------------|-------------------------------|
| <b>Name</b>      | Class display name            |
| <b>Code</b>      | Short identifier              |
| <b>Capacity</b>  | Maximum student enrollment    |
| <b>Fee Group</b> | Optional fee group assignment |

## 8.3 Courses & Subjects

- **Courses** — Programs or tracks (e.g., “General Science”, “Business”)
- **Subjects** — Individual academic subjects (e.g., “Mathematics”, “English”)
- Courses map to classes via `ClassCourses`
- Subjects map to courses via `CourseSubjects`

## 8.4 Exam Categories

**Path:** Academic > Exam Categories

Define the types of assessments:

- End of Term Exam
- Midterm Exam
- Weekly Test
- Class Exercise
- Project Work

## 8.5 Mark Types

**Path:** Academic > Grade Book > Mark Types

Configure assessment categories with weight percentages:

| Example Type | Weight |
|--------------|--------|
| Class Test   | 20%    |
| Midterm Exam | 30%    |
| End of Term  | 50%    |

**Important:** Ensure mark type weights sum to 100% per term for accurate grade computation.

# 9. Billing & Fee Structure Setup

Setting up billing requires a **bottom-up approach**: create components first, then group them, then assign to classes.

## 9.1 Setup Order

Step 1: Fee Components → Individual fee items (Tuition, Books, etc.)  
Step 2: Fee Groups → Bundle components with amounts  
Step 3: Class Fee Groups → Assign fee groups to classes  
Step 4: Class Fees → Link classes to fee/admission groups  
Step 5: Billing Cycles → Define billing periods

## 9.2 Fee Components

**Path:** Billing > Fee Components

Fee components are the atomic billing units:

| Example   | Description           |
|-----------|-----------------------|
| Tuition   | Core tuition fee      |
| Books     | Textbook charges      |
| Uniform   | School uniform fee    |
| Lunch     | Meal plan             |
| Transport | Bus service           |
| Lab Fee   | Science lab materials |

## 9.3 Fee Groups

**Path:** Billing > Fee Groups

Fee groups bundle fee components with specific amounts. Each group has a **type** that determines when it's charged:

| Type             | Code | Frequency              |
|------------------|------|------------------------|
| <b>Admission</b> | 1    | One-time at enrollment |
| <b>Terminal</b>  | 2    | Once per term          |
| <b>Monthly</b>   | 3    | Every month            |
| <b>Weekly</b>    | 4    | Every week             |
| <b>Daily</b>     | 5    | Every day              |

### Creating a fee group:

1. Create the group with a code (up to 20 characters), description, type, and status.
2. Open the group details.
3. Add fees: select a fee component and enter the amount.
4. The **Total** column on the index page shows the sum of all fees in each group.

**Duplicating:** Use the **Duplicate** action to clone a fee group — useful for creating similar structures across terms.

## 9.4 Class Fee Groups

**Path:** Billing > Class Fee Groups

Assign fee groups to each class. Each class supports up to five assignments:

| Assignment          | Required | Description                    |
|---------------------|----------|--------------------------------|
| Admission Fee Group | Yes      | One-time fees for new students |
| Terminal Fee Group  | Yes      | Per-term recurring fees        |
| Monthly Fee Group   | No       | Monthly recurring fees         |
| Weekly Fee Group    | No       | Weekly recurring fees          |
| Daily Fee Group     | No       | Daily recurring fees           |

## 9.5 Class Fees

**Path:** Billing > Class Fees

Bridge table linking classes to specific fee groups and admission groups. The system prevents duplicate combinations (same class + fee group).

## 9.6 Billing Cycles

**Path:** Billing > Billings

Create billing cycles to organize when student fees are generated:

| Field      | Description                         |
|------------|-------------------------------------|
| Name       | Cycle name (e.g., "Term 1 Billing") |
| Term       | Associated academic term            |
| Date Range | Start and end dates                 |

## 9.7 Student Subscriptions

**Path:** Billing > Subscriptions

Manage additional service subscriptions (e.g., SMS notification plans) on a per-student basis.

# 10. Settings & Lookup Data

## 10.1 Schools

**Path:** Settings > Schools

Manage the list of external schools — used in student educational history (previous schools attended).

## 10.2 Holidays

**Path:** Settings > Holidays

Configure school holidays and closures. Holidays affect:

- Attendance tracking (school days vs. holidays)
- Academic calendar calculations

## 10.3 Incident Types

**Path:** Settings > Incidents

Define the categories of disciplinary incidents:

- Tardiness
- Uniform Violation
- Fighting
- Vandalism
- Academic Dishonesty
- etc.

## 10.4 Seeded Lookup Data

The following lookup data is automatically seeded on first startup and can be managed through their respective areas:

| Lookup                | Used In                                                |
|-----------------------|--------------------------------------------------------|
| <b>Nationalities</b>  | Student admissions                                     |
| <b>Religions</b>      | Student admissions                                     |
| <b>Relationships</b>  | Guardian information (Parent, Guardian, Sibling, etc.) |
| <b>Occupations</b>    | Guardian information                                   |
| <b>Qualifications</b> | Instructor profiles                                    |
| <b>Positions</b>      | Instructor job titles                                  |

# 11. Reports Administration

## 11.1 Available Reports

The system provides 12 exportable reports organized into four categories:

### Student Reports:

- Student Enrollment — Counts by class, term, status
- Student Class List — Roster by class
- Student Demographics — Breakdown by gender, nationality

### Academic Reports:

- Attendance Summary — Statistics by class and date range
- Student Performance — Individual marks by class, term, subject
- Class Performance Summary — Class-wide averages per term

### Financial Reports:

- Fee Collection — Payment records by date and mode
- Outstanding Fees — Unpaid balances by class and term
- Fee Group Summary — Totals by fee group and term

### Staff Reports:

- Instructor List — Staff roster by status
- Instructor Workload — Class/subject load per instructor
- Disciplinary Issues — Records by class and date range

## 11.2 Report Permissions

| Permission     | Access                            |
|----------------|-----------------------------------|
| reports.view   | View report pages and filter data |
| reports.export | Export reports to PDF             |

## 11.3 PDF Branding

All exported PDFs include:

- **Report title and description** (upper left)
- **Tenant logo** (upper right) — sourced from the tenant's LogoUrl
- **Fallback logo** — wwwroot/branding/logo.png when no tenant logo is configured
- **Footer** — Page numbering
- **Row limit** — Maximum 500 rows per export to prevent performance issues

## 11.4 Customizing the Fallback Logo

Replace src/SMSAPP.Mvc/wwwroot/branding/logo.png with your organization's logo. Recommended dimensions: 200x100 pixels, PNG format.

## 12. Database Management

### 12.1 Connection String

Configure in appsettings.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=<host>;Database=<db>;User ID=<user>;Password=<pass>;Encri
```

### 12.2 Migrations

Entity Framework Core manages the database schema. Migrations are applied automatically on startup.

**Manual migration commands** (if needed):

```
# Create a new migration
dotnet ef migrations add <MigrationName> \
  --project src/SMSAPP.Data \
  --startup-project src/SMSAPP.Api
```

```
# Apply pending migrations
dotnet ef database update \
  --project src/SMSAPP.Data \
  --startup-project src/SMSAPP.Api
```

```
# Generate SQL script (for DBA review)
dotnet ef migrations script \
  --project src/SMSAPP.Data \
  --startup-project src/SMSAPP.Api
```

```
# Revert last migration (if not yet applied)
dotnet ef migrations remove \
  --project src/SMSAPP.Data \
  --startup-project src/SMSAPP.Api
```

**Important:** Always use SMSAPP.Api as the startup project for EF commands. Never manually edit generated migration files.

### 12.3 Number Formatting

Auto-generated numbers follow these formats:

| Entity              | Format                                            | Example    |
|---------------------|---------------------------------------------------|------------|
| Admission Number    | ADM-{Sequence:000000}                             | ADM-000001 |
| Registration Number | REG-{Sequence:000000}                             | REG-000001 |
| Student Number      | {FirstInitial}{LastInitial}-<br>{Sequence:000000} | JD-000001  |

### 12.4 Backup Recommendations

- Schedule daily SQL Server backups
- Keep at least 7 days of point-in-time restore capability
- Test restore procedures quarterly
- Back up wwwroot/uploads/ (profile images, documents) separately

## 13. Background Services

### 13.1 Student Registration Migration Service

**Service:** StudentRegistrationMigrationBackgroundService

This background service processes the async migration of student registrations to active student records:

1. A user triggers **Sync to Student** on a registration
2. The registration ID is queued in StudentRegistrationMigrationQueue (in-memory)
3. The background service dequeues and processes it via IAdmissionDataMigrationService
4. A Student record is created with an auto-generated student number

**Monitoring:** The service runs continuously. If the application restarts, any in-flight items in the queue are lost (in-memory queue). For production, consider implementing a persistent queue or retry mechanism.

## 13.2 Database Initialization Service

On application startup:

1. InitializeSmsAppDatabaseAsync() applies pending migrations
2. EnsureAdmissionLookupDataAsync() seeds lookup tables
3. BootstrapAdminSeeder creates the admin account (if not exists)
4. SmsAppSampleDataSeeder seeds sample data (development only)

# 14. Security & Authentication

## 14.1 Authentication Flow

1. User submits credentials via the login form
2. ApplicationAuthenticationService validates the password using a secure hasher
3. On success, a claims identity is created with:
  - Username (smsapp:username)
  - Permissions (smsapp:permission) — one claim per assigned permission
  - Tenant ID (smsapp:tenant\_id)
  - Tenant Name (smsapp:tenant\_name)
  - Super Admin flag (smsapp:is\_super\_admin)
  - Profile Image URL (smsapp:profile\_image\_url)
  - Must Change Password flag (smsapp:must\_change\_password)
4. A cookie is issued with 8-hour sliding expiration

## 14.2 Password Change Enforcement

The MustChangePassword middleware intercepts every request:

- If the user has the smsapp:must\_change\_password claim, they are redirected to /Account/ChangePassword
- This applies to: new users, password resets, bootstrap admin first login

## 14.3 Authorization Policies

Authorization is enforced at the controller and action level using [Authorize(Policy = ...)]:

```
// Require read access
[Authorize(Policy = ApplicationPermissions.StudentsView)]

// Require write access
[Authorize(Policy = ApplicationPermissions.StudentsManage)]

// Super admin only
[Authorize(Policy = "SuperAdmin")]
```

## 14.4 Security Best Practices

| Area               | Recommendation                                                                               |
|--------------------|----------------------------------------------------------------------------------------------|
| Passwords          | Enforce strong password policies; change bootstrap admin credentials immediately             |
| HTTPS              | Always deploy with HTTPS in production                                                       |
| Connection Strings | Store securely using environment variables or Azure Key Vault — not in source control        |
| Session            | 8-hour sliding expiration; users should log out on shared computers                          |
| Permissions        | Follow principle of least privilege — grant only needed permissions                          |
| Super Admin        | Limit super admin accounts to infrastructure administrators only                             |
| Uploads            | Profile images are stored in wwwroot/uploads/profile-images/ — ensure proper file validation |

## 14.5 Multi-Tenant Data Isolation

- Every database entity includes a TenantId column
- All queries are automatically filtered by the current tenant context
- Super admins bypass tenant filtering when no specific tenant is selected
- Cross-tenant data access is prevented at the data layer

---

## 15. Troubleshooting

### 15.1 Common Issues

#### Application won't start — port in use:

```
# Find process on port 5125
lsof -i :5125
```

```
# Kill the process
kill -9 <PID>
```

#### Database connection fails:

- Verify connection string in appsettings.json
- Ensure SQL Server is running and accessible
- Check firewall rules for the database port (default 1433)
- Verify user credentials and database permissions

#### Migrations fail:

```
# Check pending migrations
dotnet ef migrations list --project src/SMSAPP.Data --startup-project src/SMSAPP.Api

# Generate SQL script for manual review
dotnet ef migrations script --project src/SMSAPP.Data --startup-project src/SMSAPP.Api
```

#### User can't access a module:

- Check the user's role assignments in **Security > Users**
- Verify the role has the required permissions in **Security > Roles**
- User must log out and back in for permission changes to take effect

#### PDF export not working:

- Ensure the user has reports.export permission
- Check that wwwroot/branding/logo.png exists (for fallback)
- Verify QuestPDF Community license is configured in Program.cs

#### Tenant not resolving:

- Verify the subdomain matches the tenant record exactly
- Ensure the tenant is in **Active** status
- Check DNS configuration points the subdomain to the application

### 15.2 Logs

ASP.NET Core logs are output to the console by default. For production, configure structured logging:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.EntityFrameworkCore": "Warning"
    }
  }
}
```

### 15.3 Health Checks

Monitor these endpoints and services:

- **Application** — HTTP GET to the root URL should return 200
- **Database** — EF Core connectivity (startup will fail if DB is unreachable)
- **Background Services** — Check application logs for registration migration service activity

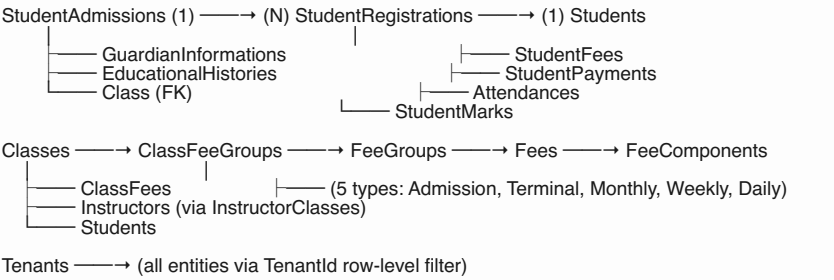
### 15.4 Performance Tuning

| Area                 | Action                                                                                       |
|----------------------|----------------------------------------------------------------------------------------------|
| Database             | Add indexes for frequently filtered columns (TenantId, ClassId, TermId)                      |
| Reports              | Row limit is 500 — increase MaxRows in ReportsController if needed, but monitor memory usage |
| Response Compression | Brotli + Gzip enabled on API; static files compressed in MVC                                 |
| Caching              | Consider adding output caching for read-heavy report pages                                   |
| Connection Pooling   | SQL Server connection pooling is enabled by default via EF Core                              |

## Appendix A: Permission Reference

| Code                 | Display Name         | Description                                         |
|----------------------|----------------------|-----------------------------------------------------|
| users.view           | View Users           | View user accounts                                  |
| users.manage         | Manage Users         | Create, edit, reset, unlock users                   |
| roles.view           | View Roles           | View role definitions                               |
| roles.manage         | Manage Roles         | Create and edit roles                               |
| permissions.view     | View Permissions     | View permission catalog                             |
| permissions.manage   | Manage Permissions   | Manage permission definitions                       |
| students.view        | View Students        | View admissions, registrations, students            |
| students.manage      | Manage Students      | Create/edit admissions, registrations, promotions   |
| academics.view       | View Academics       | View classes, attendance, grade book, lesson plans  |
| academics.manage     | Manage Academics     | Create/edit classes, attendance, marks, lessons     |
| staff.view           | View Staff           | View instructors and disciplinary records           |
| staff.manage         | Manage Staff         | Create/edit instructors and discipline              |
| billing.view         | View Billing         | View fee groups, components, student fees, payments |
| billing.manage       | Manage Billing       | Create/edit fees, record payments                   |
| reports.view         | View Reports         | View report pages                                   |
| reports.export       | Export Reports       | Export reports to PDF                               |
| communication.view   | View Communication   | View notifications and SMS queue                    |
| communication.manage | Manage Communication | Send notifications                                  |
| settings.view        | View Settings        | View system configuration                           |
| settings.manage      | Manage Settings      | Edit schools, holidays, incident types              |
| tenants.view         | View Tenants         | View tenant list (Super Admin)                      |
| tenants.manage       | Manage Tenants       | Edit tenants (Super Admin)                          |
| tenants.create       | Create Tenants       | Onboard new tenants (Super Admin)                   |

## Appendix B: Entity Relationship Summary



## Appendix C: Default Ports

| Application | HTTP | HTTPS |
|-------------|------|-------|
| MVC (Web)   | 5125 | 7218  |
| API         | 5228 | 7220  |

---

© 2026 School Management System. All rights reserved.