Database Schema

Ape Framework - Database Schema

Introduction

This document provides the complete database schema for the Ape Framework. The SQL script below can be executed against a fresh SQL Server database to create all required tables, indexes, and seed data without using Entity Framework migrations.

Database Requirements

- **SQL Server:** 2019 or later (or Azure SQL Database)
- **Collation:** SQL_Latin1_General_CP1_CI_AS (default)

Quick Setup

1. Create a new database in SQL Server
2. Run the SQL script in the next section
3. Update your connection string or environment variables
4. The default admin account will be created:
    - **Email:** admin@admin.com
    - **Password:** Admin123!

Complete SQL Schema Script

```sql
-- ================================================
-- APE FRAMEWORK DATABASE SCHEMA
-- Version: 1.0.0
-- Compatible with: SQL Server 2019+, Azure SQL
-- ================================================

-- ================================================
-- SECTION 1: ASP.NET IDENTITY TABLES
-- ================================================

-- AspNetRoles - Stores roles (Admin, Member)
CREATE TABLE [dbo].[AspNetRoles] (
    [Id] NVARCHAR(450) NOT NULL,
    [Name] NVARCHAR(256) NULL,
    [NormalizedName] NVARCHAR(256) NULL,
    [ConcurrencyStamp] NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC)
);
GO

CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex]
    ON [dbo].[AspNetRoles]([NormalizedName] ASC)
    WHERE ([NormalizedName] IS NOT NULL);
GO

-- AspNetUsers - Stores user accounts
CREATE TABLE [dbo].[AspNetUsers] (
    [Id] NVARCHAR(450) NOT NULL,
    [UserName] NVARCHAR(256) NULL,
    [NormalizedUserName] NVARCHAR(256) NULL,
    [Email] NVARCHAR(256) NULL,
    [NormalizedEmail] NVARCHAR(256) NULL,
    [EmailConfirmed] BIT NOT NULL
```

```
    [EmailConfirmed] BIT NOT NULL,
    [PasswordHash] NVARCHAR(MAX) NULL,
    [SecurityStamp] NVARCHAR(MAX) NULL,
    [ConcurrencyStamp] NVARCHAR(MAX) NULL,
    [PhoneNumber] NVARCHAR(MAX) NULL,
    [PhoneNumberConfirmed] BIT NOT NULL,
    [TwoFactorEnabled] BIT NOT NULL,
    [LockoutEnd] DATETIMEOFFSET(7) NULL,
    [LockoutEnabled] BIT NOT NULL,
    [AccessFailedCount] INT NOT NULL,
    CONSTRAINT [PK_AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)
);
GO

CREATE NONCLUSTERED INDEX [EmailIndex]
    ON [dbo].[AspNetUsers]([NormalizedEmail] ASC);
GO

CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex]
    ON [dbo].[AspNetUsers]([NormalizedUserName] ASC)
    WHERE ([NormalizedUserName] IS NOT NULL);
GO

-- AspNetRoleClaims - Claims associated with roles
CREATE TABLE [dbo].[AspNetRoleClaims] (
    [Id] INT IDENTITY(1,1) NOT NULL,
    [RoleId] NVARCHAR(450) NOT NULL,
    [ClaimType] NVARCHAR(MAX) NULL,
    [ClaimValue] NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_AspNetRoleClaims] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_AspNetRoleClaims_AspNetRoles_RoleId] FOREIGN KEY ([Role
        REFERENCES [dbo].[AspNetRoles] ([Id]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_AspNetRoleClaims_RoleId]
    ON [dbo].[AspNetRoleClaims]([RoleId] ASC);
GO

-- AspNetUserClaims - Claims associated with users
CREATE TABLE [dbo].[AspNetUserClaims] (
    [Id] INT IDENTITY(1,1) NOT NULL,
    [UserId] NVARCHAR(450) NOT NULL,
    [ClaimType] NVARCHAR(MAX) NULL,
    [ClaimValue] NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_AspNetUserClaims] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_AspNetUserClaims_AspNetUsers_UserId] FOREIGN KEY ([User
        REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_AspNetUserClaims_UserId]
    ON [dbo].[AspNetUserClaims]([UserId] ASC);
GO

-- AspNetUserLogins - External login providers
CREATE TABLE [dbo].[AspNetUserLogins] (
    [LoginProvider] NVARCHAR(128) NOT NULL,
    [ProviderKey] NVARCHAR(128) NOT NULL,
    [ProviderDisplayName] NVARCHAR(MAX) NULL,
    [UserId] NVARCHAR(450) NOT NULL,
    CONSTRAINT [PK_AspNetUserLogins] PRIMARY KEY CLUSTERED ([LoginProvider
    CONSTRAINT [FK_AspNetUserLogins_AspNetUsers_UserId] FOREIGN KEY ([User
        REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_AspNetUserLogins_UserId]
    ON [dbo].[AspNetUserLogins]([UserId] ASC);
GO

-- AspNetUserRoles - User-Role assignments
CREATE TABLE [dbo].[AspNetUserRoles] (
    [UserId] NVARCHAR(450) NOT NULL,
    [RoleId] NVARCHAR(450) NOT NULL,
    CONSTRAINT [PK_AspNetUserRoles] PRIMARY KEY CLUSTERED ([UserId] ASC, [
    CONSTRAINT [FK_AspNetUserRoles_AspNetRoles_RoleId] FOREIGN KEY ([RoleI
```

```
        REFERENCES [dbo].[AspNetRoles] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_AspNetUserRoles_AspNetUsers_UserId] FOREIGN KEY ([UserI
        REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_AspNetUserRoles_RoleId]
    ON [dbo].[AspNetUserRoles]([RoleId] ASC);
GO

-- AspNetUserTokens - Authentication tokens
CREATE TABLE [dbo].[AspNetUserTokens] (
    [UserId] NVARCHAR(450) NOT NULL,
    [LoginProvider] NVARCHAR(128) NOT NULL,
    [Name] NVARCHAR(128) NOT NULL,
    [Value] NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_AspNetUserTokens] PRIMARY KEY CLUSTERED ([UserId] ASC,
    CONSTRAINT [FK_AspNetUserTokens_AspNetUsers_UserId] FOREIGN KEY ([User
        REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

-- ====================================================
-- SECTION 2: APPLICATION TABLES
-- ====================================================

-- UserProfiles - Extended user information
CREATE TABLE [dbo].[UserProfiles] (
    [UserId] NVARCHAR(450) NOT NULL,
    [FirstName] NVARCHAR(MAX) NULL,
    [MiddleName] NVARCHAR(MAX) NULL,
    [LastName] NVARCHAR(MAX) NULL,
    [Birthday] DATETIME2(7) NULL,
    [Anniversary] DATETIME2(7) NULL,
    [AddressLine1] NVARCHAR(MAX) NULL,
    [AddressLine2] NVARCHAR(MAX) NULL,
    [City] NVARCHAR(MAX) NULL,
    [State] NVARCHAR(MAX) NULL,
    [ZipCode] NVARCHAR(MAX) NULL,
    [IsMinor] NVARCHAR(MAX) NULL,
    [AgeVerified] NVARCHAR(MAX) NULL,
    [ParentalConsentDate] DATETIME2(7) NULL,
    [HomePhoneNumber] NVARCHAR(MAX) NULL,
    [LastLogin] DATETIME2(7) NULL,
    [LastActivity] DATETIME2(7) NULL,
    [IsActive] BIT NOT NULL DEFAULT 1,
    [DeactivatedDate] DATETIME2(7) NULL,
    [DeactivatedBy] NVARCHAR(100) NULL,
    [DeactivationReason] NVARCHAR(500) NULL,
    CONSTRAINT [PK_UserProfiles] PRIMARY KEY CLUSTERED ([UserId] ASC),
    CONSTRAINT [FK_UserProfiles_AspNetUsers_UserId] FOREIGN KEY ([UserId])
        REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

-- SystemCredentials - Encrypted credential storage
CREATE TABLE [dbo].[SystemCredentials] (
    [CredentialID] INT IDENTITY(1,1) NOT NULL,
    [CredentialKey] NVARCHAR(100) NOT NULL,
    [CredentialName] NVARCHAR(200) NOT NULL,
    [Category] NVARCHAR(50) NOT NULL,
    [EncryptedValue] VARBINARY(MAX) NOT NULL,
    [Description] NVARCHAR(MAX) NULL,
    [IsActive] BIT NOT NULL DEFAULT 1,
    [CreatedDate] DATETIME2(7) NOT NULL DEFAULT GETUTCDATE(),
    [CreatedBy] NVARCHAR(100) NULL,
    [UpdatedDate] DATETIME2(7) NULL,
    [UpdatedBy] NVARCHAR(100) NULL,
    CONSTRAINT [PK_SystemCredentials] PRIMARY KEY CLUSTERED ([CredentialID
);
GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_SystemCredentials_CredentialKey]
    ON [dbo].[SystemCredentials]([CredentialKey] ASC);
GO
```

```
-- CredentialAuditLogs - Audit trail for credential access
CREATE TABLE [dbo].[CredentialAuditLogs] (
    [AuditID] INT IDENTITY(1,1) NOT NULL,
    [CredentialID] INT NOT NULL,
    [CredentialKey] NVARCHAR(100) NOT NULL,
    [Action] NVARCHAR(50) NOT NULL,
    [ActionDetails] NVARCHAR(MAX) NULL,
    [ActionBy] NVARCHAR(100) NULL,
    [ActionDate] DATETIME2(7) NOT NULL DEFAULT GETUTCDATE(),
    [IPAddress] NVARCHAR(50) NULL,
    [UserAgent] NVARCHAR(500) NULL,
    [Success] BIT NULL,
    [ErrorMessage] NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_CredentialAuditLogs] PRIMARY KEY CLUSTERED ([AuditID] A
    CONSTRAINT [FK_CredentialAuditLogs_SystemCredentials_CredentialID] FOR
        REFERENCES [dbo].[SystemCredentials] ([CredentialID]) ON DELETE CA
);
GO

CREATE NONCLUSTERED INDEX [IX_CredentialAuditLogs_CredentialID]
    ON [dbo].[CredentialAuditLogs]([CredentialID] ASC);
GO

CREATE NONCLUSTERED INDEX [IX_CredentialAuditLogs_ActionDate]
    ON [dbo].[CredentialAuditLogs]([ActionDate] DESC);
GO

-- EmailLogs - Email transmission history
CREATE TABLE [dbo].[EmailLogs] (
    [Id] INT IDENTITY(1,1) NOT NULL,
    [Timestamp] DATETIME2(7) NOT NULL,
    [ToEmail] NVARCHAR(255) NOT NULL,
    [FromEmail] NVARCHAR(255) NULL,
    [Subject] NVARCHAR(500) NOT NULL,
    [Success] BIT NOT NULL,
    [Details] NVARCHAR(1000) NULL,
    [EmailServer] NVARCHAR(50) NOT NULL DEFAULT 'Unknown',
    [MessageId] NVARCHAR(100) NULL,
    [RetryCount] INT NULL DEFAULT 0,
    [SentBy] NVARCHAR(100) NULL,
    [EmailType] NVARCHAR(50) NULL,
    [CreatedDate] DATETIME2(7) NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT [PK_EmailLogs] PRIMARY KEY CLUSTERED ([Id] ASC)
);
GO

CREATE NONCLUSTERED INDEX [IX_EmailLogs_Timestamp]
    ON [dbo].[EmailLogs]([Timestamp] DESC);
GO

CREATE NONCLUSTERED INDEX [IX_EmailLogs_ToEmail]
    ON [dbo].[EmailLogs]([ToEmail] ASC);
GO

-- SystemSettings - Application configuration
CREATE TABLE [dbo].[SystemSettings] (
    [Id] INT IDENTITY(1,1) NOT NULL,
    [SettingKey] NVARCHAR(100) NOT NULL,
    [SettingValue] NVARCHAR(1000) NOT NULL,
    [Description] NVARCHAR(500) NULL,
    [CreatedDate] DATETIME2(7) NOT NULL DEFAULT GETUTCDATE(),
    [UpdatedDate] DATETIME2(7) NOT NULL DEFAULT GETUTCDATE(),
    [UpdatedBy] NVARCHAR(100) NULL,
    CONSTRAINT [PK_SystemSettings] PRIMARY KEY CLUSTERED ([Id] ASC)
);
GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_SystemSettings_SettingKey]
    ON [dbo].[SystemSettings]([SettingKey] ASC);
GO

-- PDFCategories - Document library folders
CREATE TABLE [dbo].[PDFCategories] (
    [CategoryID] INT IDENTITY(1,1) NOT NULL,
    [CategoryName] NVARCHAR(MAX) NOT NULL,
    [SortOrder] INT NOT NULL,
    [ParentCategoryID] INT NULL,
```

```
    [ParentCategoryID] INT NULL,
    [AccessLevel] INT NOT NULL DEFAULT 0,  -- 0=Member, 1=Admin
    CONSTRAINT [PK_PDFCategories] PRIMARY KEY CLUSTERED ([CategoryID] ASC)
    CONSTRAINT [FK_PDFCategories_PDFCategories_ParentCategoryID] FOREIGN K
        REFERENCES [dbo].[PDFCategories] ([CategoryID]) ON DELETE NO ACTIO
);
GO

CREATE NONCLUSTERED INDEX [IX_PDFCategories_ParentCategoryID]
    ON [dbo].[PDFCategories]([ParentCategoryID] ASC);
GO

-- CategoryFiles - Document files
CREATE TABLE [dbo].[CategoryFiles] (
    [FileID] INT IDENTITY(1,1) NOT NULL,
    [CategoryID] INT NOT NULL,
    [FileName] NVARCHAR(MAX) NOT NULL,
    [SortOrder] INT NOT NULL DEFAULT 0,
    [Description] NVARCHAR(500) NULL,
    [UploadedDate] DATETIME2(7) NULL,
    [UploadedBy] NVARCHAR(256) NULL,
    CONSTRAINT [PK_CategoryFiles] PRIMARY KEY CLUSTERED ([FileID] ASC),
    CONSTRAINT [FK_CategoryFiles_PDFCategories_CategoryID] FOREIGN KEY ([C
        REFERENCES [dbo].[PDFCategories] ([CategoryID]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_CategoryFiles_CategoryID]
    ON [dbo].[CategoryFiles]([CategoryID] ASC);
GO

-- GalleryCategories - Image gallery categories
CREATE TABLE [dbo].[GalleryCategories] (
    [CategoryID] INT IDENTITY(1,1) NOT NULL,
    [CategoryName] NVARCHAR(MAX) NOT NULL,
    [SortOrder] INT NOT NULL,
    [ParentCategoryID] INT NULL,
    [AccessLevel] INT NOT NULL DEFAULT 0,  -- 0=Member, 1=Admin
    [Description] NVARCHAR(500) NULL,
    CONSTRAINT [PK_GalleryCategories] PRIMARY KEY CLUSTERED ([CategoryID]
    CONSTRAINT [FK_GalleryCategories_GalleryCategories_ParentCategoryID] F
        REFERENCES [dbo].[GalleryCategories] ([CategoryID]) ON DELETE NO A
);
GO

CREATE NONCLUSTERED INDEX [IX_GalleryCategories_ParentCategoryID]
    ON [dbo].[GalleryCategories]([ParentCategoryID] ASC);
GO

-- GalleryImages - Gallery images
CREATE TABLE [dbo].[GalleryImages] (
    [ImageID] INT IDENTITY(1,1) NOT NULL,
    [CategoryID] INT NOT NULL,
    [FileName] NVARCHAR(MAX) NOT NULL,
    [OriginalFileName] NVARCHAR(500) NULL,
    [SortOrder] INT NOT NULL DEFAULT 0,
    [Description] NVARCHAR(500) NULL,
    [UploadedDate] DATETIME2(7) NULL,
    [UploadedBy] NVARCHAR(256) NULL,
    CONSTRAINT [PK_GalleryImages] PRIMARY KEY CLUSTERED ([ImageID] ASC),
    CONSTRAINT [FK_GalleryImages_GalleryCategories_CategoryID] FOREIGN KEY
        REFERENCES [dbo].[GalleryCategories] ([CategoryID]) ON DELETE CASC
);
GO

CREATE NONCLUSTERED INDEX [IX_GalleryImages_CategoryID]
    ON [dbo].[GalleryImages]([CategoryID] ASC);
GO

-- LinkCategories - Link directory categories
CREATE TABLE [dbo].[LinkCategories] (
    [CategoryID] INT IDENTITY(1,1) NOT NULL,
    [CategoryName] NVARCHAR(MAX) NOT NULL,
    [SortOrder] INT NOT NULL,
    [IsAdminOnly] BIT NOT NULL DEFAULT 0,
    CONSTRAINT [PK_LinkCategories] PRIMARY KEY CLUSTERED ([CategoryID] ASC
);
```

```
);
GO

-- CategoryLinks - Individual links
CREATE TABLE [dbo].[CategoryLinks] (
    [LinkID] INT IDENTITY(1,1) NOT NULL,
    [CategoryID] INT NOT NULL,
    [LinkName] NVARCHAR(MAX) NOT NULL,
    [LinkUrl] NVARCHAR(MAX) NOT NULL,
    [SortOrder] INT NOT NULL DEFAULT 0,
    CONSTRAINT [PK_CategoryLinks] PRIMARY KEY CLUSTERED ([LinkID] ASC),
    CONSTRAINT [FK_CategoryLinks_LinkCategories_CategoryID] FOREIGN KEY ([
        REFERENCES [dbo].[LinkCategories] ([CategoryID]) ON DELETE CASCADE
);
GO

CREATE NONCLUSTERED INDEX [IX_CategoryLinks_CategoryID]
    ON [dbo].[CategoryLinks]([CategoryID] ASC);
GO

-- EF Migrations History (optional - for tracking if migrations are later
CREATE TABLE [dbo].[__EFMigrationsHistory] (
    [MigrationId] NVARCHAR(150) NOT NULL,
    [ProductVersion] NVARCHAR(32) NOT NULL,
    CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY CLUSTERED ([Migratio
);
GO

-- ======================================================
-- SECTION 3: SEED DATA
-- ======================================================

-- Insert default roles
INSERT INTO [dbo].[AspNetRoles] ([Id], [Name], [NormalizedName], [Concurre
VALUES
    (NEWID(), 'Admin', 'ADMIN', NEWID()),
    (NEWID(), 'Member', 'MEMBER', NEWID());
GO

-- Insert default admin user
-- Password: Admin123! (hashed with ASP.NET Identity default hasher)
-- NOTE: This hash was generated for "Admin123!" - you may need to regener
--       if your Identity configuration differs
DECLARE @AdminId NVARCHAR(450) = NEWID();
DECLARE @AdminRoleId NVARCHAR(450);

SELECT @AdminRoleId = [Id] FROM [dbo].[AspNetRoles] WHERE [NormalizedName]

-- Insert admin user
INSERT INTO [dbo].[AspNetUsers] (
    [Id], [UserName], [NormalizedUserName], [Email], [NormalizedEmail],
    [EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp],
    [PhoneNumber], [PhoneNumberConfirmed], [TwoFactorEnabled],
    [LockoutEnd], [LockoutEnabled], [AccessFailedCount]
)
VALUES (
    @AdminId,
    'admin@admin.com',
    'ADMIN@ADMIN.COM',
    'admin@admin.com',
    'ADMIN@ADMIN.COM',
    1,  -- EmailConfirmed = true
    -- Password hash for "Admin123!" using ASP.NET Identity V3
    'AQAAAAIAAYagAAAAEKxJEVpMvP5BZv8zVQmZAJUFMFT7fMjFHVP2qF0MnKmB+5FxON5S3
    NEWID(),  -- SecurityStamp
    NEWID(),  -- ConcurrencyStamp
    NULL,     -- PhoneNumber
    0,        -- PhoneNumberConfirmed
    0,        -- TwoFactorEnabled
    NULL,     -- LockoutEnd
    1,        -- LockoutEnabled
    0         -- AccessFailedCount
);

-- Assign Admin role to admin user
INSERT INTO [dbo].[AspNetUserRoles] ([UserId], [RoleId])
VALUES (@AdminId, @AdminRoleId);
```

```sql
-- Create UserProfile for admin
INSERT INTO [dbo].[UserProfiles] ([UserId], [IsActive], [LastLogin])
VALUES (@AdminId, 1, GETUTCDATE());

GO

-- Insert default system settings
INSERT INTO [dbo].[SystemSettings] ([SettingKey], [SettingValue], [Descrip
VALUES
    ('SiteName', 'Ape Framework', 'The display name of the site', GETUTCDA
    ('ContactFormEmails', '', 'Comma-separated list of email addresses to
GO

-- Mark migrations as applied (so EF doesn't try to re-run them)
INSERT INTO [dbo].[__EFMigrationsHistory] ([MigrationId], [ProductVersion]
VALUES
    ('00000000000000_CreateIdentitySchema', '10.0.0'),
    ('20260126025147_AddUserProfiles', '10.0.0'),
    ('20260126114355_AddSystemCredentials', '10.0.0'),
    ('20260126215448_AddEmailLogsAndSystemSettings', '10.0.0'),
    ('20260127015551_AddDocumentLibrary', '10.0.0'),
    ('20260127035245_AddLastActivity', '10.0.0'),
    ('20260127181801_AddImageGallery', '10.0.0'),
    ('20260127205603_AddMoreLinks', '10.0.0');
GO

PRINT 'Database schema created successfully!';
PRINT 'Default admin account: admin@admin.com / Admin123!';
PRINT 'IMPORTANT: Change the admin password immediately after first login!
GO
```

Table Summary

Identity Tables

Table	Purpose
AspNetUsers	User accounts with email, password hash, 2FA settings
AspNetRoles	Role definitions (Admin, Member)
AspNetUserRoles	User-to-role assignments
AspNetUserClaims	Claims associated with users
AspNetRoleClaims	Claims associated with roles
AspNetUserLogins	External login provider data
AspNetUserTokens	Authentication tokens

Application Tables

Table	Purpose
UserProfiles	Extended user data (name, address, activity tracking)
SystemCredentials	Encrypted credential storage
CredentialAuditLogs	Audit trail for credential access
EmailLogs	Email transmission history
SystemSettings	Key-value application settings
PDFCategories	Document library folder hierarchy
CategoryFiles	PDF documents in folders
GalleryCategories	Image gallery category hierarchy
GalleryImages	Images in categories
LinkCategories	Link directory categories
CategoryLinks	Links in categories

Access Level Values

The `AccessLevel` column in PDFCategories and GalleryCategories uses these values:

Value	Level	Description
0	Member	Visible to all authenticated users
1	Admin	Visible only to administrators

Important Notes

Password Hash

The password hash in the seed data is for "Admin123!" using ASP.NET Identity's default hasher. If your application uses different Identity settings, you may need to:

1. Let the application create the admin user automatically on first run, OR
2. Generate a new hash using the same hasher configuration

Master Encryption Key

The SystemCredentials table stores encrypted values. Before adding credentials:

1. Set the `MASTER_CREDENTIAL_KEY_ILLUSTRATE` environment variable
2. The key must be at least 32 characters
3. All credential values are encrypted with AES-256

File Storage

This script creates the database schema only. You also need:

- `/ProtectedFiles/` directory for PDF documents
- `/wwwroot/Galleries/` directory for gallery images

Ensure the application has write permissions to these directories.

Verification Queries

After running the script, verify with these queries:

```
-- Check tables were created
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
ORDER BY TABLE_NAME;

-- Check roles exist
SELECT * FROM AspNetRoles;

-- Check admin user exists
SELECT Id, Email, EmailConfirmed FROM AspNetUsers;

-- Check admin has Admin role
SELECT u.Email, r.Name as Role
FROM AspNetUsers u
JOIN AspNetUserRoles ur ON u.Id = ur.UserId
JOIN AspNetRoles r ON ur.RoleId = r.Id;

-- Check system settings
SELECT * FROM SystemSettings;
```

Version: 1.0.0 **Framework:** Ape Framework **Site:** https://Illustrate.net