using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

//public class GradingBody

//{

// public int Id { get; set; }

// public string Name { get; set; } // e.g., "Zimbabwe Education Board", "Cambridge"

// public string Country { get; set; }

// public string Description { get; set; }

// public DateTime CreatedAt { get; set; }

// public ICollection<GradingScheme> GradingSchemes { get; set; } = new List<GradingScheme>();

// public ICollection<School> Schools { get; set; } = new List<School>();

//}

namespace SchoolManagement.Core.Entities

{

public class GradingScheme : BaseEntity

{

[Required]

[StringLength(100)]

public string Name { get; set; } = string.Empty;

[StringLength(500)]

public string Description { get; set; } = string.Empty;

[StringLength(50)]

public string Country { get; set; } = string.Empty;

public bool IsActive { get; set; } = true;

// Navigation Properties

public virtual ICollection<GradingScale> GradingScales { get; set; } = new List<GradingScale>();

public virtual ICollection<School> Schools { get; set; } = new List<School>();

}

}

// Models/Entities/GradingScale.cs

namespace SchoolManagement.Core.Entities

{

public class GradingScale : BaseEntity

{

public Guid GradingSchemeId { get; set; }

public virtual GradingScheme GradingScheme { get; set; } = null!;

[StringLength(10)]

public string Symbol { get; set; } = string.Empty; // A, B, C, D, etc.

public int Unit { get; set; } // 1, 2, 3, 4, etc.

public decimal MinPercentage { get; set; }

public decimal MaxPercentage { get; set; }

[StringLength(100)]

public string Description { get; set; } = string.Empty; // Excellent, Good, Fair, etc.

public int SortOrder { get; set; }

}

}

}

namespace SmartXul.Api.Models.Exams

{

public class Exam : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid GradeId { get; set; }

public virtual Grade Grade { get; set; } = null!;

public Guid TermId { get; set; }

public virtual Term Term { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public Guid CreatedByTeacherId { get; set; }

public virtual Teacher CreatedByTeacher { get; set; } = null!;

[Required]

[StringLength(200)]

public string Title { get; set; } = string.Empty;

[StringLength(1000)]

public string Description { get; set; } = string.Empty;

public ExamType Type { get; set; }

public DateTime ExamDate { get; set; }

public TimeSpan Duration { get; set; }

public TimeSpan StartTime { get; set; }

public TimeSpan EndTime { get; set; }

public decimal TotalMarks { get; set; }

public decimal PassingMarks { get; set; }

[StringLength(100)]

public string Venue { get; set; } = string.Empty;

public ExamStatus Status { get; set; } = ExamStatus.Draft;

public bool IsPublished { get; set; } = false;

public DateTime? PublishedAt { get; set; }

public bool AllowHandwritingRecognition { get; set; } = false;

public bool IsOnline { get; set; } = false;

public bool RequiresSupervision { get; set; } = true;

[StringLength(500)]

public string Instructions { get; set; } = string.Empty;

public string MaterialsRequired { get; set; } = string.Empty; // JSON list

// Navigation Properties

public virtual ICollection<ExamQuestion> Questions { get; set; } = new List<ExamQuestion>();

public virtual ICollection<ExamRegistration> Registrations { get; set; } = new List<ExamRegistration>();

public virtual ICollection<ExamSession> Sessions { get; set; } = new List<ExamSession>();

public virtual ICollection<ExamResult> Results { get; set; } = new List<ExamResult>();

public virtual ICollection<ExamSupervisor> Supervisors { get; set; } = new List<ExamSupervisor>();

}

}

namespace SmartXul.Api.Models.Exams

{

public class ExamAnswer : BaseEntity

{

public Guid ExamSessionId { get; set; }

public virtual ExamSession ExamSession { get; set; } = null!;

public Guid ExamQuestionId { get; set; }

public virtual ExamQuestion ExamQuestion { get; set; } = null!;

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public string AnswerText { get; set; } = string.Empty;

// For file uploads or handwritten answers

public string AttachmentPath { get; set; } = string.Empty;

public string HandwritingImagePath { get; set; } = string.Empty;

// AI Recognition results

public string RecognizedText { get; set; } = string.Empty;

public decimal AiConfidenceScore { get; set; }

public bool IsAiProcessed { get; set; } = false;

// Marking

public decimal? MarksObtained { get; set; }

public bool IsMarked { get; set; } = false;

public Guid? MarkedByTeacherId { get; set; }

public virtual Teacher? MarkedByTeacher { get; set; }

public DateTime? MarkedAt { get; set; }

public string TeacherFeedback { get; set; } = string.Empty;

// Answer metadata

public DateTime AnsweredAt { get; set; } = DateTime.UtcNow;

public TimeSpan TimeSpent { get; set; }

public int AttemptNumber { get; set; } = 1;

public bool IsFlagged { get; set; } = false;

public string FlagReason { get; set; } = string.Empty;

}

}

using SmartXul.Shared.Enums;

using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models.Exams

{

public class ExamIncident : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public Guid? StudentId { get; set; }

public virtual Student? Student { get; set; }

public Guid ReportedByTeacherId { get; set; }

public virtual Teacher ReportedByTeacher { get; set; } = null!;

public IncidentType Type { get; set; }

public IncidentSeverity Severity { get; set; }

[Required]

public string Description { get; set; } = string.Empty;

public DateTime IncidentTime { get; set; }

public string Location { get; set; } = string.Empty;

public string ActionTaken { get; set; } = string.Empty;

public bool RequiresFollowUp { get; set; } = false;

public string FollowUpNotes { get; set; } = string.Empty;

public string AttachmentsPath { get; set; } = string.Empty; // Evidence photos/videos

public IncidentStatus Status { get; set; } = IncidentStatus.Open;

public DateTime? ResolvedAt { get; set; }

public Guid? ResolvedByTeacherId { get; set; }

public virtual Teacher? ResolvedByTeacher { get; set; }

}

}

using SmartXul.Shared.Enums;

using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models.Exams

{

public class ExamQuestion : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public int QuestionNumber { get; set; }

[Required]

public string QuestionText { get; set; } = string.Empty;

public QuestionType Type { get; set; }

public decimal Marks { get; set; }

public bool IsRequired { get; set; } = true;

public TimeSpan? TimeLimit { get; set; }

// For multiple choice questions

public string Options { get; set; } = string.Empty; // JSON array

public string CorrectAnswer { get; set; } = string.Empty;

// For essay/written questions

public int? WordLimit { get; set; }

public string SampleAnswer { get; set; } = string.Empty;

public string MarkingRubric { get; set; } = string.Empty; // JSON

// For handwriting recognition

public bool EnableHandwritingRecognition { get; set; } = false;

public string ExpectedKeywords { get; set; } = string.Empty; // JSON array

// Media attachments

public string AttachmentPath { get; set; } = string.Empty;

public int SortOrder { get; set; }

// Navigation Properties

public virtual ICollection<ExamAnswer> Answers { get; set; } = new List<ExamAnswer>();

}

}

using SmartXul.Shared.Enums;

using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models.Exams

{

public class ExamRegistration : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public DateTime RegistrationDate { get; set; } = DateTime.UtcNow;

public RegistrationStatus Status { get; set; } = RegistrationStatus.Registered;

public bool IsPresent { get; set; } = false;

public DateTime? CheckInTime { get; set; }

public DateTime? CheckOutTime { get; set; }

[StringLength(100)]

public string SeatNumber { get; set; } = string.Empty;

public bool RequiresSpecialAccommodation { get; set; } = false;

public string SpecialAccommodations { get; set; } = string.Empty;

[StringLength(500)]

public string Remarks { get; set; } = string.Empty;

}

}

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models.Exams

{

public class ExamResult : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid ExamSessionId { get; set; }

public virtual ExamSession ExamSession { get; set; } = null!;

public decimal TotalMarks { get; set; }

public decimal MarksObtained { get; set; }

public decimal Percentage { get; set; }

public string Grade { get; set; } = string.Empty;

public int GradeUnit { get; set; }

public ResultStatus Status { get; set; } = ResultStatus.Pending;

public bool IsPassed { get; set; }

public string Remarks { get; set; } = string.Empty;

// Marking details

public bool IsFullyMarked { get; set; } = false;

public int QuestionsMarked { get; set; }

public int TotalQuestions { get; set; }

public Guid? MarkedByTeacherId { get; set; }

public virtual Teacher? MarkedByTeacher { get; set; }

public DateTime? MarkedAt { get; set; }

// Result publication

public bool IsPublished { get; set; } = false;

public DateTime? PublishedAt { get; set; }

// Analytics

public int Rank { get; set; }

public decimal? ClassAverage { get; set; }

public decimal? HighestScore { get; set; }

public decimal? LowestScore { get; set; }

public string DetailedAnalysis { get; set; } = string.Empty; // JSON

}

}

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models.Exams

{

public class ExamSession : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public DateTime StartTime { get; set; }

public DateTime? EndTime { get; set; }

public TimeSpan ActualDuration { get; set; }

public SessionStatus Status { get; set; } = SessionStatus.NotStarted;

public bool IsSubmitted { get; set; } = false;

public DateTime? SubmittedAt { get; set; }

public bool IsAutoSubmitted { get; set; } = false;

public int TotalQuestions { get; set; }

public int AnsweredQuestions { get; set; }

// Proctoring data

public string DeviceInfo { get; set; } = string.Empty; // JSON

public string IPAddress { get; set; } = string.Empty;

public string BrowserInfo { get; set; } = string.Empty;

// Security flags

public bool HasViolations { get; set; } = false;

public string ViolationDetails { get; set; } = string.Empty; // JSON

public decimal? Score { get; set; }

public decimal? Percentage { get; set; }

// Navigation Properties

public virtual ICollection<ExamAnswer> Answers { get; set; } = new List<ExamAnswer>();

public virtual ICollection<ExamSessionLog> SessionLogs { get; set; } = new List<ExamSessionLog>();

}

}

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models.Exams

{

public class ExamSessionLog : BaseEntity

{

public Guid ExamSessionId { get; set; }

public virtual ExamSession ExamSession { get; set; } = null!;

public LogType Type { get; set; }

public string Event { get; set; } = string.Empty;

public string Details { get; set; } = string.Empty; // JSON

public DateTime Timestamp { get; set; } = DateTime.UtcNow;

public string IPAddress { get; set; } = string.Empty;

public string UserAgent { get; set; } = string.Empty;

public bool IsSuspicious { get; set; } = false;

public string SuspicionReason { get; set; } = string.Empty;

}

}

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models.Exams

{

public class ExamSupervisor : BaseEntity

{

public Guid ExamId { get; set; }

public virtual Exam Exam { get; set; } = null!;

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public SupervisorRole Role { get; set; }

public DateTime AssignedAt { get; set; } = DateTime.UtcNow;

public bool IsPresent { get; set; } = false;

public DateTime? CheckInTime { get; set; }

public DateTime? CheckOutTime { get; set; }

public string Responsibilities { get; set; } = string.Empty;

public string Notes { get; set; } = string.Empty;

// Navigation Properties

public virtual ICollection<ExamIncident> ReportedIncidents { get; set; } = new List<ExamIncident>();

}

}

using SmartXul.Shared;

using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Attendance : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public Guid TermId { get; set; }

public virtual Term Term { get; set; } = null!;

public DateTime Date { get; set; }

public AttendanceStatus Status { get; set; }

[StringLength(200)]

public string Remarks { get; set; } = string.Empty;

}

}

using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public abstract class BaseEntity

{

[Key]

public Guid Id { get; set; } = Guid.NewGuid();

public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

public DateTime UpdatedAt { get; set; } = DateTime.UtcNow;

public string CreatedBy { get; set; } = string.Empty;

public string UpdatedBy { get; set; } = string.Empty;

public bool IsDeleted { get; set; } = false;

}

}

namespace SmartXul.Api.Models

{

public class ClassTeacher : BaseEntity

{

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public Guid GradeId { get; set; }

public virtual Grade Grade { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public bool IsActive { get; set; } = true;

}

}

using System.ComponentModel.DataAnnotations;

using System.Xml;

namespace SmartXul.Api.Models

{

public class Grade : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

[Required]

[StringLength(50)]

public string Name { get; set; } = string.Empty; // Grade 1, Form 1, Year 7, etc.

[StringLength(200)]

public string Description { get; set; } = string.Empty;

public int Level { get; set; } // 1, 2, 3, etc. for ordering

public int MaxStudents { get; set; } = 50;

// Navigation Properties

public virtual ICollection<Student> Students { get; set; } = new List<Student>();

public virtual ICollection<GradeSubject> GradeSubjects { get; set; } = new List<GradeSubject>();

public virtual ICollection<ClassTeacher> ClassTeachers { get; set; } = new List<ClassTeacher>();

public virtual ICollection<Timetable> Timetables { get; set; } = new List<Timetable>();

}

}namespace SmartXul.Api.Models

{

public class GradeSubject : BaseEntity

{

public Guid GradeId { get; set; }

public virtual Grade Grade { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public bool IsCompulsory { get; set; } = true;

public int SortOrder { get; set; }

}

}using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class HandwritingCharacterData : BaseEntity

{

public Guid HandwritingTrainingDataId { get; set; }

public virtual HandwritingTrainingData HandwritingTrainingData { get; set; } = null!;

[Required]

public string Character { get; set; } = string.Empty; // The actual character (A, B, 1, 2, etc.)

[Required]

public string ImageSegmentPath { get; set; } = string.Empty; // Path to individual character image

// Bounding box coordinates

public int X { get; set; }

public int Y { get; set; }

public int Width { get; set; }

public int Height { get; set; }

public decimal ConfidenceScore { get; set; }

public bool IsVerified { get; set; } = false;

// Feature vectors for ML model

public string FeatureVector { get; set; } = string.Empty; // JSON array of features

public int SequenceOrder { get; set; } // Order in the original text

}

}using System.ComponentModel.DataAnnotations;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class HandwritingModel : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

[Required]

[StringLength(100)]

public string ModelName { get; set; } = string.Empty;

[Required]

[StringLength(50)]

public string Version { get; set; } = string.Empty;

public string ModelFilePath { get; set; } = string.Empty; // Local device model path

public string CloudModelPath { get; set; } = string.Empty; // Cloud backup path

public ModelType Type { get; set; }

public decimal Accuracy { get; set; }

public int TrainingDataCount { get; set; }

public bool IsActive { get; set; } = true;

public bool IsDeployedLocally { get; set; } = false;

public bool IsDeployedCloud { get; set; } = false;

public DateTime TrainingStarted { get; set; }

public DateTime? TrainingCompleted { get; set; }

public DateTime? LastUsed { get; set; }

// Model metadata

public string TrainingParameters { get; set; } = string.Empty; // JSON

public string PerformanceMetrics { get; set; } = string.Empty; // JSON

// Navigation Properties

public virtual ICollection<HandwritingRecognitionResult> RecognitionResults { get; set; } = new List<HandwritingRecognitionResult>();

}

}using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class HandwritingRecognitionResult : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid? SubjectId { get; set; }

public virtual Subject? Subject { get; set; }

public Guid? ResultId { get; set; }

public virtual Result? Result { get; set; }

public Guid HandwritingModelId { get; set; }

public virtual HandwritingModel HandwritingModel { get; set; } = null!;

[Required]

public string OriginalImagePath { get; set; } = string.Empty;

[Required]

public string RecognizedText { get; set; } = string.Empty;

public decimal OverallConfidence { get; set; }

public bool ProcessedLocally { get; set; } = true;

public bool ProcessedInCloud { get; set; } = false;

public TimeSpan ProcessingTime { get; set; }

public string CharacterConfidences { get; set; } = string.Empty; // JSON array

public string AlternativeTexts { get; set; } = string.Empty; // JSON array of alternatives

// For answer verification

public string ExpectedAnswer { get; set; } = string.Empty;

public bool IsCorrect { get; set; }

public decimal PartialCreditScore { get; set; }

public DateTime ProcessedAt { get; set; } = DateTime.UtcNow;

}

}using System.ComponentModel.DataAnnotations;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class HandwritingTrainingData : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid? TeacherId { get; set; }

public virtual Teacher? Teacher { get; set; }

public TrainingDataType DataType { get; set; }

[Required]

public string OriginalContent { get; set; } = string.Empty; // What the student was supposed to write

[Required]

public string ImagePath { get; set; } = string.Empty; // Path to handwritten image

public string ProcessedText { get; set; } = string.Empty; // AI interpreted text

public string VerifiedText { get; set; } = string.Empty; // Teacher verified text

public decimal ConfidenceScore { get; set; }

public TrainingStatus Status { get; set; } = TrainingStatus.Pending;

public bool IsVerified { get; set; } = false;

public bool IsTrainingComplete { get; set; } = false;

public DateTime CapturedAt { get; set; } = DateTime.UtcNow;

public DateTime? VerifiedAt { get; set; }

public DateTime? TrainedAt { get; set; }

// Metadata for training

public string BoundingBoxes { get; set; } = string.Empty; // JSON array of character bounding boxes

//public string CharacterData { get; set; } = string.Empty; // JSON array of individual character data

public string ModelVersion { get; set; } = string.Empty;

// Navigation Properties

public virtual ICollection<HandwritingCharacterData> CharacterData { get; set; } = new List<HandwritingCharacterData>();

}

}

using System.ComponentModel.DataAnnotations;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class ModelTrainingJob : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

[Required]

[StringLength(100)]

public string JobName { get; set; } = string.Empty;

public JobType Type { get; set; }

public JobStatus Status { get; set; } = JobStatus.Queued;

public int TotalDataPoints { get; set; }

public int ProcessedDataPoints { get; set; }

public decimal Progress { get; set; }

public DateTime QueuedAt { get; set; } = DateTime.UtcNow;

public DateTime? StartedAt { get; set; }

public DateTime? CompletedAt { get; set; }

public string ErrorMessage { get; set; } = string.Empty;

public string ResultPath { get; set; } = string.Empty;

// Training parameters

public string TrainingParameters { get; set; } = string.Empty; // JSON

// Navigation Properties

public virtual HandwritingModel? ResultingModel { get; set; }

}

} using System.ComponentModel.DataAnnotations;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class Notification : BaseEntity

{

public Guid? ParentId { get; set; }

public virtual Parent? Parent { get; set; }

public Guid? StudentId { get; set; }

public virtual Student? Student { get; set; }

public Guid? TeacherId { get; set; }

public virtual Teacher? Teacher { get; set; }

[Required]

[StringLength(200)]

public string Title { get; set; } = string.Empty;

[Required]

public string Message { get; set; } = string.Empty;

public NotificationType Type { get; set; }

public NotificationChannel Channel { get; set; }

public bool IsRead { get; set; } = false;

public bool IsSent { get; set; } = false;

public DateTime? SentAt { get; set; }

public DateTime? ReadAt { get; set; }

[StringLength(100)]

public string ExternalId { get; set; } = string.Empty; // For WhatsApp, SMS tracking

public string Metadata { get; set; } = string.Empty; // JSON for additional data

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Parent : BaseEntity

{

[Required]

[StringLength(100)]

public string FirstName { get; set; } = string.Empty;

[Required]

[StringLength(100)]

public string LastName { get; set; } = string.Empty;

[EmailAddress]

[StringLength(100)]

public string Email { get; set; } = string.Empty;

[StringLength(20)]

public string PhoneNumber { get; set; } = string.Empty;

[StringLength(20)]

public string WhatsAppNumber { get; set; } = string.Empty;

[StringLength(200)]

public string Address { get; set; } = string.Empty;

[StringLength(50)]

public string Relationship { get; set; } = string.Empty; // Father, Mother, Guardian

[StringLength(100)]

public string Occupation { get; set; } = string.Empty;

// Navigation Properties

public virtual ICollection<Student> Students { get; set; } = new List<Student>();

public virtual ICollection<Notification> Notifications { get; set; } = new List<Notification>();

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Result : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public Guid TermId { get; set; }

public virtual Term Term { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public decimal Score { get; set; }

public decimal TotalMarks { get; set; }

public decimal Percentage { get; set; }

[StringLength(10)]

public string Grade { get; set; } = string.Empty; // A, B, C, etc.

public int GradeUnit { get; set; } // 1, 2, 3, etc.

[StringLength(50)]

public string AssessmentType { get; set; } = string.Empty; // Test, Exam, Assignment

[StringLength(500)]

public string Comments { get; set; } = string.Empty;

public DateTime AssessmentDate { get; set; }

// AI Answer Verification

public bool IsAiVerified { get; set; } = false;

public decimal AiConfidenceScore { get; set; }

public string AiProcessedAnswers { get; set; } = string.Empty; // JSON

public object PartialCreditScore { get; internal set; }

}

} using System.ComponentModel.DataAnnotations;

using System.Diagnostics;

using System.Xml;

using SmartXul.Api.Models.SchoolManagement.Core.Entities;

namespace SmartXul.Api.Models

{

public class School : BaseEntity

{

[Required]

[StringLength(200)]

public string Name { get; set; } = string.Empty;

[StringLength(500)]

public string Address { get; set; } = string.Empty;

[StringLength(20)]

public string PhoneNumber { get; set; } = string.Empty;

[EmailAddress]

[StringLength(100)]

public string Email { get; set; } = string.Empty;

[StringLength(100)]

public string Website { get; set; } = string.Empty;

[StringLength(50)]

public string RegistrationNumber { get; set; } = string.Empty;

public string Logo { get; set; } = string.Empty;

// Navigation Properties

public virtual ICollection<Teacher> Teachers { get; set; } = new List<Teacher>();

public virtual ICollection<Student> Students { get; set; } = new List<Student>();

public virtual ICollection<Grade> Grades { get; set; } = new List<Grade>();

public virtual ICollection<Subject> Subjects { get; set; } = new List<Subject>();

public virtual ICollection<SchoolYear> SchoolYears { get; set; } = new List<SchoolYear>();

public virtual ICollection<Timetable> Timetables { get; set; } = new List<Timetable>();

// Grading Scheme Reference

public Guid? GradingSchemeId { get; set; }

public virtual GradingScheme? GradingScheme { get; set; }

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class SchoolYear : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

[Required] [StringLength(50)] public string Name { get; set; } = string.Empty; // 2024/2025

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public bool IsActive { get; set; } = true;

public bool IsCurrent { get; set; } = false;

// Navigation Properties

public virtual ICollection<Term> Terms { get; set; } = new List<Term>();

public virtual ICollection<SubjectTeacher> SubjectTeachers { get; set; } = new List<SubjectTeacher>();

public virtual ICollection<StudentSubject> StudentSubjects { get; set; } = new List<StudentSubject>();

public virtual ICollection<ClassTeacher> ClassTeachers { get; set; } = new List<ClassTeacher>();

public virtual ICollection<Result> Results { get; set; } = new List<Result>();

}

public class Term : BaseEntity

{

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

[Required] [StringLength(50)] public string Name { get; set; } = string.Empty; // Term 1, Semester 1, etc.

public int TermNumber { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public bool IsActive { get; set; } = true;

public bool IsCurrent { get; set; } = false;

// Navigation Properties

public virtual ICollection<Result> Results { get; set; } = new List<Result>();

public virtual ICollection<Attendance> AttendanceRecords { get; set; } = new List<Attendance>();

}

} using System.ComponentModel.DataAnnotations;

using System.Diagnostics;

namespace SmartXul.Api.Models

{

public class Student : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

[Required]

[StringLength(100)]

public string FirstName { get; set; } = string.Empty;

[Required]

[StringLength(100)]

public string LastName { get; set; } = string.Empty;

[StringLength(50)]

public string StudentNumber { get; set; } = string.Empty;

public DateTime DateOfBirth { get; set; }

[StringLength(10)]

public string Gender { get; set; } = string.Empty;

[StringLength(200)]

public string Address { get; set; } = string.Empty;

public string ProfilePicture { get; set; } = string.Empty;

// Current Grade

public Guid CurrentGradeId { get; set; }

public virtual Grade CurrentGrade { get; set; } = null!;

// Navigation Properties

public virtual ICollection<Parent> Parents { get; set; } = new List<Parent>();

public virtual ICollection<StudentSubject> StudentSubjects { get; set; } = new List<StudentSubject>();

public virtual ICollection<Attendance> AttendanceRecords { get; set; } = new List<Attendance>();

public virtual ICollection<Result> Results { get; set; } = new List<Result>();

public virtual ICollection<HandwritingTrainingData> HandwritingTrainingData { get; set; } = new List<HandwritingTrainingData>();

public virtual ICollection<StudentTimetable> StudentTimetables { get; set; } = new List<StudentTimetable>();

}

} namespace SmartXul.Api.Models

{

public class StudentSubject : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public DateTime EnrollmentDate { get; set; } = DateTime.UtcNow;

public bool IsActive { get; set; } = true;

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class StudentTimetable : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid TimetableSlotId { get; set; }

public virtual TimetableSlot TimetableSlot { get; set; } = null!;

public bool IsActive { get; set; } = true;

// For individual student customizations

[StringLength(200)]

public string CustomNotes { get; set; } = string.Empty;

}

} using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class StudentTrainingSession : BaseEntity

{

public Guid StudentId { get; set; }

public virtual Student Student { get; set; } = null!;

public Guid TrainingExerciseId { get; set; }

public virtual TrainingExercise TrainingExercise { get; set; } = null!;

public DateTime StartedAt { get; set; }

public DateTime? CompletedAt { get; set; }

public SessionStatus Status { get; set; } = SessionStatus.InProgress;

public int TotalItems { get; set; }

public int CompletedItems { get; set; }

public int CorrectItems { get; set; }

public decimal OverallAccuracy { get; set; }

public TimeSpan TotalTime { get; set; }

// Navigation Properties

public virtual ICollection<HandwritingTrainingData> TrainingData { get; set; } = new List<HandwritingTrainingData>();

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Subject : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

[Required]

[StringLength(100)]

public string Name { get; set; } = string.Empty;

[StringLength(10)]

public string Code { get; set; } = string.Empty;

[StringLength(500)]

public string Description { get; set; } = string.Empty;

public int Credits { get; set; } = 1;

public bool IsActive { get; set; } = true;

// Navigation Properties

public virtual ICollection<GradeSubject> GradeSubjects { get; set; } = new List<GradeSubject>();

public virtual ICollection<SubjectTeacher> SubjectTeachers { get; set; } = new List<SubjectTeacher>();

public virtual ICollection<StudentSubject> StudentSubjects { get; set; } = new List<StudentSubject>();

public virtual ICollection<Result> Results { get; set; } = new List<Result>();

public virtual ICollection<TimetableSlot> TimetableSlots { get; set; } = new List<TimetableSlot>();

}

} namespace SmartXul.Api.Models

{

public class SubjectTeacher : BaseEntity

{

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public Guid GradeId { get; set; }

public virtual Grade Grade { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public bool IsActive { get; set; } = true;

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Teacher : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

[Required]

[StringLength(100)]

public string FirstName { get; set; } = string.Empty;

[Required]

[StringLength(100)]

public string LastName { get; set; } = string.Empty;

[EmailAddress]

[StringLength(100)]

public string Email { get; set; } = string.Empty;

[StringLength(20)]

public string PhoneNumber { get; set; } = string.Empty;

[StringLength(200)]

public string Address { get; set; } = string.Empty;

[StringLength(50)]

public string EmployeeId { get; set; } = string.Empty;

public DateTime DateOfBirth { get; set; }

public DateTime HireDate { get; set; }

[StringLength(100)]

public string Qualification { get; set; } = string.Empty;

public string ProfilePicture { get; set; } = string.Empty;

// Navigation Properties

public virtual ICollection<SubjectTeacher> SubjectTeachers { get; set; } = new List<SubjectTeacher>();

public virtual ICollection<ClassTeacher> ClassTeachers { get; set; } = new List<ClassTeacher>();

public virtual ICollection<Attendance> AttendanceRecords { get; set; } = new List<Attendance>();

public virtual ICollection<Result> Results { get; set; } = new List<Result>();

public virtual ICollection<TimetableSlot> TimetableSlots { get; set; } = new List<TimetableSlot>();

}

} using System.ComponentModel.DataAnnotations;

namespace SmartXul.Api.Models

{

public class Timetable : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

public Guid GradeId { get; set; }

public virtual Grade Grade { get; set; } = null!;

public Guid SchoolYearId { get; set; }

public virtual SchoolYear SchoolYear { get; set; } = null!;

public Guid TermId { get; set; }

public virtual Term Term { get; set; } = null!;

[Required]

[StringLength(100)]

public string Name { get; set; } = string.Empty;

public bool IsActive { get; set; } = true;

public bool IsGenerated { get; set; } = false;

public DateTime GeneratedAt { get; set; }

// Navigation Properties

public virtual ICollection<TimetableSlot> TimetableSlots { get; set; } = new List<TimetableSlot>();

}

public class TimetableSlot : BaseEntity

{

public Guid TimetableId { get; set; }

public virtual Timetable Timetable { get; set; } = null!;

public Guid SubjectId { get; set; }

public virtual Subject Subject { get; set; } = null!;

public Guid TeacherId { get; set; }

public virtual Teacher Teacher { get; set; } = null!;

public DayOfWeek DayOfWeek { get; set; }

public TimeSpan StartTime { get; set; }

public TimeSpan EndTime { get; set; }

[StringLength(50)]

public string Room { get; set; } = string.Empty;

public int PeriodNumber { get; set; }

[StringLength(200)]

public string Notes { get; set; } = string.Empty;

}

} using System.ComponentModel.DataAnnotations;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Models

{

public class TrainingExercise : BaseEntity

{

public Guid SchoolId { get; set; }

public virtual School School { get; set; } = null!;

public Guid? GradeId { get; set; }

public virtual Grade? Grade { get; set; }

public Guid? SubjectId { get; set; }

public virtual Subject? Subject { get; set; }

[Required]

[StringLength(200)]

public string Title { get; set; } = string.Empty;

[StringLength(1000)]

public string Description { get; set; } = string.Empty;

public ExerciseType Type { get; set; }

public DifficultyLevel Difficulty { get; set; }

[Required]

public string Content { get; set; } = string.Empty; // JSON containing exercise data

public bool IsActive { get; set; } = true;

public bool IsSystemGenerated { get; set; } = false;

public int EstimatedMinutes { get; set; } = 10;

// Navigation Properties

public virtual ICollection<StudentTrainingSession> StudentSessions { get; set; } = new List<StudentTrainingSession>();

}

} using Microsoft.EntityFrameworkCore.ChangeTracking;

using Microsoft.EntityFrameworkCore;

using SmartXul.Api.Models;

using SmartXul.Api.Models.SchoolManagement.Core.Entities;

using System.Linq.Expressions;

using SmartXul.Api.Models.Exams;

namespace SmartXul.Api

{

public class SmartXulDbContext : DbContext

{

public SmartXulDbContext(DbContextOptions<SmartXulDbContext> options)

: base(options)

{

}

// Core Entities

public DbSet<School> Schools { get; set; }

public DbSet<GradingScheme> GradingSchemes { get; set; }

public DbSet<GradingScale> GradingScales { get; set; }

public DbSet<Teacher> Teachers { get; set; }

public DbSet<Student> Students { get; set; }

public DbSet<Parent> Parents { get; set; }

public DbSet<Grade> Grades { get; set; }

public DbSet<Subject> Subjects { get; set; }

public DbSet<SchoolYear> SchoolYears { get; set; }

public DbSet<Term> Terms { get; set; }

// Relationship Entities

public DbSet<GradeSubject> GradeSubjects { get; set; }

public DbSet<SubjectTeacher> SubjectTeachers { get; set; }

public DbSet<StudentSubject> StudentSubjects { get; set; }

public DbSet<ClassTeacher> ClassTeachers { get; set; }

// Academic Records

public DbSet<Attendance> Attendances { get; set; }

public DbSet<Result> Results { get; set; }

public DbSet<Notification> Notifications { get; set; }

// Timetable

public DbSet<Timetable> Timetables { get; set; }

public DbSet<TimetableSlot> TimetableSlots { get; set; }

public DbSet<StudentTimetable> StudentTimetables { get; set; }

// AI Handwriting Recognition

public DbSet<HandwritingTrainingData> HandwritingTrainingData { get; set; }

public DbSet<HandwritingCharacterData> HandwritingCharacterData { get; set; }

public DbSet<HandwritingModel> HandwritingModels { get; set; }

public DbSet<HandwritingRecognitionResult> HandwritingRecognitionResults { get; set; }

public DbSet<TrainingExercise> TrainingExercises { get; set; }

public DbSet<StudentTrainingSession> StudentTrainingSessions { get; set; }

public DbSet<ModelTrainingJob> ModelTrainingJobs { get; set; }

// Exams

public DbSet<Exam> Exams { get; set; }

public DbSet<ExamQuestion> ExamQuestions { get; set; }

public DbSet<ExamAnswer> ExamAnswers { get; set; }

public DbSet<ExamRegistration> ExamRegistrations { get; set; }

public DbSet<ExamSession> ExamSessions { get; set; }

public DbSet<ExamResult> ExamResults { get; set; }

public DbSet<ExamSupervisor> ExamSupervisors { get; set; }

public DbSet<ExamIncident> ExamIncidents { get; set; }

public DbSet<ExamSessionLog> ExamSessionLogs { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

// Configure entity relationships and constraints

ConfigureSchoolEntities(modelBuilder);

ConfigureGradingEntities(modelBuilder);

ConfigureUserEntities(modelBuilder);

ConfigureAcademicEntities(modelBuilder);

ConfigureTimetableEntities(modelBuilder);

ConfigureAIEntities(modelBuilder);

ConfigureIndexes(modelBuilder);

ConfigureSoftDelete(modelBuilder);

}

private void ConfigureSchoolEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<School>(entity =>

{

entity.HasIndex(e => e.RegistrationNumber).IsUnique();

entity.HasIndex(e => e.Email).IsUnique();

entity.Property(e => e.Name).IsRequired();

});

modelBuilder.Entity<SchoolYear>(entity =>

{

entity.HasOne(sy => sy.School)

.WithMany(s => s.SchoolYears)

.HasForeignKey(sy => sy.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.SchoolId, e.Name }).IsUnique();

entity.HasIndex(e => new { e.SchoolId, e.IsCurrent });

});

modelBuilder.Entity<Term>(entity =>

{

entity.HasOne(t => t.SchoolYear)

.WithMany(sy => sy.Terms)

.HasForeignKey(t => t.SchoolYearId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.SchoolYearId, e.TermNumber }).IsUnique();

});

}

private void ConfigureGradingEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<GradingScheme>(entity =>

{

entity.HasIndex(e => new { e.Name, e.Country }).IsUnique();

entity.Property(e => e.Name).IsRequired();

});

modelBuilder.Entity<GradingScale>(entity =>

{

entity.HasOne(gs => gs.GradingScheme)

.WithMany(g => g.GradingScales)

.HasForeignKey(gs => gs.GradingSchemeId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.GradingSchemeId, e.Symbol }).IsUnique();

entity.HasIndex(e => new { e.GradingSchemeId, e.Unit }).IsUnique();

entity.Property(e => e.MinPercentage).HasColumnType("decimal(5,2)");

entity.Property(e => e.MaxPercentage).HasColumnType("decimal(5,2)");

});

modelBuilder.Entity<School>()

.HasOne(s => s.GradingScheme)

.WithMany(gs => gs.Schools)

.HasForeignKey(s => s.GradingSchemeId)

.OnDelete(DeleteBehavior.SetNull);

}

private void ConfigureUserEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Teacher>(entity =>

{

entity.HasOne(t => t.School)

.WithMany(s => s.Teachers)

.HasForeignKey(t => t.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.SchoolId, e.Email }).IsUnique();

entity.HasIndex(e => new { e.SchoolId, e.EmployeeId }).IsUnique();

entity.Property(e => e.FirstName).IsRequired();

entity.Property(e => e.LastName).IsRequired();

});

modelBuilder.Entity<Student>(entity =>

{

entity.HasOne(s => s.School)

.WithMany(sch => sch.Students)

.HasForeignKey(s => s.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(s => s.CurrentGrade)

.WithMany(g => g.Students)

.HasForeignKey(s => s.CurrentGradeId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.SchoolId, e.StudentNumber }).IsUnique();

entity.Property(e => e.FirstName).IsRequired();

entity.Property(e => e.LastName).IsRequired();

// Many-to-many relationship with Parents

entity.HasMany(s => s.Parents)

.WithMany(p => p.Students)

.UsingEntity(j => j.ToTable("StudentParents"));

});

modelBuilder.Entity<Parent>(entity =>

{

entity.HasIndex(e => e.Email).IsUnique();

entity.Property(e => e.FirstName).IsRequired();

entity.Property(e => e.LastName).IsRequired();

});

}

private void ConfigureAcademicEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Grade>(entity =>

{

entity.HasOne(g => g.School)

.WithMany(s => s.Grades)

.HasForeignKey(g => g.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.SchoolId, e.Name }).IsUnique();

entity.HasIndex(e => new { e.SchoolId, e.Level });

entity.Property(e => e.Name).IsRequired();

});

modelBuilder.Entity<Subject>(entity =>

{

entity.HasOne(s => s.School)

.WithMany(sch => sch.Subjects)

.HasForeignKey(s => s.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.SchoolId, e.Code }).IsUnique();

entity.Property(e => e.Name).IsRequired();

});

// Configure relationship entities

ConfigureRelationshipEntities(modelBuilder);

// Configure attendance

modelBuilder.Entity<Attendance>(entity =>

{

entity.HasOne(a => a.Student)

.WithMany(s => s.AttendanceRecords)

.HasForeignKey(a => a.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(a => a.Subject)

.WithMany()

.HasForeignKey(a => a.SubjectId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(a => a.Teacher)

.WithMany(t => t.AttendanceRecords)

.HasForeignKey(a => a.TeacherId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(a => a.Term)

.WithMany(t => t.AttendanceRecords)

.HasForeignKey(a => a.TermId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.StudentId, e.SubjectId, e.Date }).IsUnique();

});

// Configure results

modelBuilder.Entity<Result>(entity =>

{

entity.HasOne(r => r.Student)

.WithMany(s => s.Results)

.HasForeignKey(r => r.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(r => r.Subject)

.WithMany(s => s.Results)

.HasForeignKey(r => r.SubjectId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(r => r.Teacher)

.WithMany(t => t.Results)

.HasForeignKey(r => r.TeacherId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(r => r.Term)

.WithMany(t => t.Results)

.HasForeignKey(r => r.TermId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(r => r.SchoolYear)

.WithMany(sy => sy.Results)

.HasForeignKey(r => r.SchoolYearId)

.OnDelete(DeleteBehavior.Restrict);

entity.Property(e => e.Score).HasColumnType("decimal(6,2)");

entity.Property(e => e.TotalMarks).HasColumnType("decimal(6,2)");

entity.Property(e => e.Percentage).HasColumnType("decimal(5,2)");

entity.Property(e => e.AiConfidenceScore).HasColumnType("decimal(5,4)");

entity.Property(e => e.PartialCreditScore).HasColumnType("decimal(5,2)");

});

}

private void ConfigureRelationshipEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<GradeSubject>(entity =>

{

entity.HasOne(gs => gs.Grade)

.WithMany(g => g.GradeSubjects)

.HasForeignKey(gs => gs.GradeId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(gs => gs.Subject)

.WithMany(s => s.GradeSubjects)

.HasForeignKey(gs => gs.SubjectId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.GradeId, e.SubjectId }).IsUnique();

});

modelBuilder.Entity<SubjectTeacher>(entity =>

{

entity.HasOne(st => st.Subject)

.WithMany(s => s.SubjectTeachers)

.HasForeignKey(st => st.SubjectId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(st => st.Teacher)

.WithMany(t => t.SubjectTeachers)

.HasForeignKey(st => st.TeacherId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(st => st.Grade)

.WithMany()

.HasForeignKey(st => st.GradeId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(st => st.SchoolYear)

.WithMany(sy => sy.SubjectTeachers)

.HasForeignKey(st => st.SchoolYearId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.SubjectId, e.TeacherId, e.GradeId, e.SchoolYearId }).IsUnique();

});

modelBuilder.Entity<StudentSubject>(entity =>

{

entity.HasOne(ss => ss.Student)

.WithMany(s => s.StudentSubjects)

.HasForeignKey(ss => ss.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(ss => ss.Subject)

.WithMany(s => s.StudentSubjects)

.HasForeignKey(ss => ss.SubjectId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(ss => ss.SchoolYear)

.WithMany(sy => sy.StudentSubjects)

.HasForeignKey(ss => ss.SchoolYearId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.StudentId, e.SubjectId, e.SchoolYearId }).IsUnique();

});

modelBuilder.Entity<ClassTeacher>(entity =>

{

entity.HasOne(ct => ct.Teacher)

.WithMany(t => t.ClassTeachers)

.HasForeignKey(ct => ct.TeacherId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(ct => ct.Grade)

.WithMany(g => g.ClassTeachers)

.HasForeignKey(ct => ct.GradeId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(ct => ct.SchoolYear)

.WithMany(sy => sy.ClassTeachers)

.HasForeignKey(ct => ct.SchoolYearId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.GradeId, e.SchoolYearId }).IsUnique();

});

}

private void ConfigureTimetableEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Timetable>(entity =>

{

entity.HasOne(t => t.School)

.WithMany(s => s.Timetables)

.HasForeignKey(t => t.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(t => t.Grade)

.WithMany(g => g.Timetables)

.HasForeignKey(t => t.GradeId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(t => t.SchoolYear)

.WithMany()

.HasForeignKey(t => t.SchoolYearId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(t => t.Term)

.WithMany()

.HasForeignKey(t => t.TermId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.GradeId, e.SchoolYearId, e.TermId }).IsUnique();

});

modelBuilder.Entity<TimetableSlot>(entity =>

{

entity.HasOne(ts => ts.Timetable)

.WithMany(t => t.TimetableSlots)

.HasForeignKey(ts => ts.TimetableId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(ts => ts.Subject)

.WithMany(s => s.TimetableSlots)

.HasForeignKey(ts => ts.SubjectId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasOne(ts => ts.Teacher)

.WithMany(t => t.TimetableSlots)

.HasForeignKey(ts => ts.TeacherId)

.OnDelete(DeleteBehavior.Restrict);

entity.HasIndex(e => new { e.TimetableId, e.DayOfWeek, e.PeriodNumber }).IsUnique();

entity.HasIndex(e => new { e.TeacherId, e.DayOfWeek, e.StartTime, e.EndTime });

});

modelBuilder.Entity<StudentTimetable>(entity =>

{

entity.HasOne(st => st.Student)

.WithMany(s => s.StudentTimetables)

.HasForeignKey(st => st.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(st => st.TimetableSlot)

.WithMany()

.HasForeignKey(st => st.TimetableSlotId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.StudentId, e.TimetableSlotId }).IsUnique();

});

}

private void ConfigureAIEntities(ModelBuilder modelBuilder)

{

modelBuilder.Entity<HandwritingTrainingData>(entity =>

{

entity.HasOne(htd => htd.Student)

.WithMany(s => s.HandwritingTrainingData)

.HasForeignKey(htd => htd.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(htd => htd.Teacher)

.WithMany()

.HasForeignKey(htd => htd.TeacherId)

.OnDelete(DeleteBehavior.SetNull);

entity.HasIndex(e => new { e.StudentId, e.Status });

entity.HasIndex(e => e.IsVerified);

entity.Property(e => e.ConfidenceScore).HasColumnType("decimal(5,4)");

});

modelBuilder.Entity<HandwritingCharacterData>(entity =>

{

entity.HasOne(hcd => hcd.HandwritingTrainingData)

.WithMany(htd => htd.CharacterData)

.HasForeignKey(hcd => hcd.HandwritingTrainingDataId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.HandwritingTrainingDataId, e.SequenceOrder });

entity.Property(e => e.ConfidenceScore).HasColumnType("decimal(5,4)");

});

modelBuilder.Entity<HandwritingModel>(entity =>

{

entity.HasOne(hm => hm.Student)

.WithMany()

.HasForeignKey(hm => hm.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.StudentId, e.Version }).IsUnique();

entity.HasIndex(e => new { e.StudentId, e.IsActive });

entity.Property(e => e.Accuracy).HasColumnType("decimal(5,4)");

});

modelBuilder.Entity<HandwritingRecognitionResult>(entity =>

{

entity.HasOne(hrr => hrr.Student)

.WithMany()

.HasForeignKey(hrr => hrr.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(hrr => hrr.Subject)

.WithMany()

.HasForeignKey(hrr => hrr.SubjectId)

.OnDelete(DeleteBehavior.SetNull);

entity.HasOne(hrr => hrr.Result)

.WithMany()

.HasForeignKey(hrr => hrr.ResultId)

.OnDelete(DeleteBehavior.SetNull);

entity.HasOne(hrr => hrr.HandwritingModel)

.WithMany(hm => hm.RecognitionResults)

.HasForeignKey(hrr => hrr.HandwritingModelId)

.OnDelete(DeleteBehavior.Restrict);

entity.Property(e => e.OverallConfidence).HasColumnType("decimal(5,4)");

entity.Property(e => e.PartialCreditScore).HasColumnType("decimal(5,2)");

});

modelBuilder.Entity<TrainingExercise>(entity =>

{

entity.HasOne(te => te.School)

.WithMany()

.HasForeignKey(te => te.SchoolId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(te => te.Grade)

.WithMany()

.HasForeignKey(te => te.GradeId)

.OnDelete(DeleteBehavior.SetNull);

entity.HasOne(te => te.Subject)

.WithMany()

.HasForeignKey(te => te.SubjectId)

.OnDelete(DeleteBehavior.SetNull);

entity.HasIndex(e => new { e.SchoolId, e.Type, e.Difficulty });

});

modelBuilder.Entity<StudentTrainingSession>(entity =>

{

entity.HasOne(sts => sts.Student)

.WithMany()

.HasForeignKey(sts => sts.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasOne(sts => sts.TrainingExercise)

.WithMany(te => te.StudentSessions)

.HasForeignKey(sts => sts.TrainingExerciseId)

.OnDelete(DeleteBehavior.Cascade);

entity.Property(e => e.OverallAccuracy).HasColumnType("decimal(5,4)");

});

modelBuilder.Entity<ModelTrainingJob>(entity =>

{

entity.HasOne(mtj => mtj.Student)

.WithMany()

.HasForeignKey(mtj => mtj.StudentId)

.OnDelete(DeleteBehavior.Cascade);

entity.HasIndex(e => new { e.Status, e.QueuedAt });

entity.Property(e => e.Progress).HasColumnType("decimal(5,2)");

});

}

private void ConfigureIndexes(ModelBuilder modelBuilder)

{

// Additional performance indexes

modelBuilder.Entity<Result>()

.HasIndex(e => new { e.StudentId, e.TermId, e.SubjectId });

modelBuilder.Entity<Attendance>()

.HasIndex(e => new { e.Date, e.Status });

modelBuilder.Entity<Notification>()

.HasIndex(e => new { e.Type, e.IsSent, e.CreatedAt });

}

private void ConfigureSoftDelete(ModelBuilder modelBuilder)

{

foreach (var entityType in modelBuilder.Model.GetEntityTypes())

{

if (typeof(BaseEntity).IsAssignableFrom(entityType.ClrType))

{

// Create parameter expression for the entity type

var parameter = Expression.Parameter(entityType.ClrType, "e");

// Convert parameter to BaseEntity and access IsDeleted property

var convertExpression = Expression.Convert(parameter, typeof(BaseEntity));

var propertyAccess = Expression.Property(convertExpression, nameof(BaseEntity.IsDeleted));

// Create negation expression (!IsDeleted)

var notExpression = Expression.Not(propertyAccess);

// Build lambda expression

var lambda = Expression.Lambda(notExpression, parameter);

// Set query filter using the dynamically built expression

modelBuilder.Entity(entityType.ClrType).HasQueryFilter(lambda);

}

}

}

public override int SaveChanges()

{

UpdateTimestamps();

return base.SaveChanges();

}

public override Task<int> SaveChangesAsync(CancellationToken cancellationToken = default)

{

UpdateTimestamps();

return base.SaveChangesAsync(cancellationToken);

}

private void UpdateTimestamps()

{

var entries = ChangeTracker.Entries<BaseEntity>();

foreach (var entry in entries)

{

switch (entry.State)

{

case EntityState.Added:

entry.Entity.CreatedAt = DateTime.UtcNow;

entry.Entity.UpdatedAt = DateTime.UtcNow;

break;

case EntityState.Modified:

entry.Entity.UpdatedAt = DateTime.UtcNow;

break;

}

}

}

}

} using AutoMapper;

using SmartXul.Api.Models.SchoolManagement.Core.Entities;

using SmartXul.Api.Models;

using SmartXul.Shared.DTOs;

namespace SmartXul.Api

{

public class AutoMapperProfile : Profile

{

public AutoMapperProfile()

{

// School mappings

CreateMap<School, SchoolDto>().ReverseMap();

CreateMap<CreateSchoolDto, School>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore());

CreateMap<UpdateSchoolDto, School>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Student mappings

CreateMap<Student, StudentDto>().ReverseMap();

CreateMap<CreateStudentDto, Student>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore())

.ForMember(dest => dest.Parents, opt => opt.MapFrom(src => src.Parents));

CreateMap<UpdateStudentDto, Student>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Teacher mappings

CreateMap<Teacher, TeacherDto>().ReverseMap();

CreateMap<CreateTeacherDto, Teacher>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore());

CreateMap<UpdateTeacherDto, Teacher>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Parent mappings

CreateMap<Parent, ParentDto>().ReverseMap();

CreateMap<CreateParentDto, Parent>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore());

// Grading mappings

CreateMap<GradingScheme, GradingSchemeDto>().ReverseMap();

CreateMap<CreateGradingSchemeDto, GradingScheme>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore())

.ForMember(dest => dest.IsActive, opt => opt.MapFrom(src => true));

CreateMap<UpdateGradingSchemeDto, GradingScheme>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

CreateMap<GradingScale, GradingScaleDto>().ReverseMap();

CreateMap<CreateGradingScaleDto, GradingScale>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore());

CreateMap<UpdateGradingScaleDto, GradingScale>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Core entity mappings

CreateMap<Grade, GradeDto>().ReverseMap();

CreateMap<Subject, SubjectDto>().ReverseMap();

CreateMap<Term, TermDto>().ReverseMap();

CreateMap<SchoolYear, SchoolYearDto>().ReverseMap();

// Result mappings

CreateMap<Result, ResultDto>().ReverseMap();

CreateMap<CreateResultDto, Result>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore())

.ForMember(dest => dest.Percentage, opt => opt.Ignore())

.ForMember(dest => dest.Grade, opt => opt.Ignore())

.ForMember(dest => dest.GradeUnit, opt => opt.Ignore());

CreateMap<UpdateResultDto, Result>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Attendance mappings

CreateMap<Attendance, AttendanceDto>().ReverseMap();

CreateMap<CreateAttendanceDto, Attendance>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore());

CreateMap<UpdateAttendanceDto, Attendance>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Timetable mappings

CreateMap<Timetable, TimetableDto>().ReverseMap();

CreateMap<TimetableSlot, TimetableSlotDto>().ReverseMap();

CreateMap<UpdateTimetableSlotDto, TimetableSlot>()

.ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

// Notification mappings

CreateMap<Notification, NotificationDto>().ReverseMap();

CreateMap<CreateNotificationDto, Notification>()

.ForMember(dest => dest.Id, opt => opt.Ignore())

.ForMember(dest => dest.CreatedAt, opt => opt.Ignore())

.ForMember(dest => dest.UpdatedAt, opt => opt.Ignore())

.ForMember(dest => dest.IsRead, opt => opt.MapFrom(src => false))

.ForMember(dest => dest.IsSent, opt => opt.MapFrom(src => false));

// AI Handwriting mappings

CreateMap<HandwritingTrainingData, HandwritingTrainingDataDto>().ReverseMap();

CreateMap<HandwritingRecognitionResult, HandwritingRecognitionResultDto>()

.ForMember(d => d.AlternativeTexts, opt => opt.MapFrom(src => JsonHelpers.ToStringList(src.AlternativeTexts)))

.ForMember(d => d.CharacterConfidences, opt => opt.MapFrom(src => JsonHelpers.ToDecimalList(src.CharacterConfidences)));

CreateMap<TrainingExercise, TrainingExerciseDto>()

.ForMember(d => d.Content, opt => opt.MapFrom(src => JsonHelpers.ToExerciseContent(src.Content)));

CreateMap<StudentTrainingSession, StudentTrainingSessionDto>().ReverseMap();

CreateMap<ModelTrainingJob, ModelTrainingJobDto>().ReverseMap();

CreateMap<HandwritingModel, HandwritingModelDto>()

.ForMember(d => d.PerformanceMetrics, opt => opt.MapFrom(src => JsonHelpers.ToDictionary(src.PerformanceMetrics)));

}

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api

{

public static class JsonHelpers

{

public static Dictionary<string, object> ToDictionary(string json)

{

if (string.IsNullOrWhiteSpace(json)) return new Dictionary<string, object>();

return System.Text.Json.JsonSerializer.Deserialize<Dictionary<string, object>>(json)

?? new Dictionary<string, object>();

}

public static List<string> ToStringList(string json)

{

if (string.IsNullOrWhiteSpace(json)) return new List<string>();

return System.Text.Json.JsonSerializer.Deserialize<List<string>>(json)

?? new List<string>();

}

public static List<decimal> ToDecimalList(string json)

{

if (string.IsNullOrWhiteSpace(json)) return new List<decimal>();

return System.Text.Json.JsonSerializer.Deserialize<List<decimal>>(json)

?? new List<decimal>();

}

public static ExerciseContentDto ToExerciseContent(string json)

{

if (string.IsNullOrWhiteSpace(json)) return new ExerciseContentDto();

return System.Text.Json.JsonSerializer.Deserialize<ExerciseContentDto>(json)

?? new ExerciseContentDto();

}

}

}

Service Interfaces

using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface IAttendanceService

{

Task<AttendanceDto> MarkAttendanceAsync(CreateAttendanceDto createAttendanceDto);

Task<IEnumerable<AttendanceDto>> GetAttendanceByClassAsync(Guid gradeId, DateTime date);

Task<IEnumerable<AttendanceDto>> GetStudentAttendanceAsync(Guid studentId, Guid termId);

Task<AttendanceReportDto> GetAttendanceReportAsync(Guid studentId, Guid termId);

Task<bool> BulkMarkAttendanceAsync(List<CreateAttendanceDto> attendanceList);

Task<AttendanceDto> UpdateAttendanceAsync(Guid attendanceId, UpdateAttendanceDto updateDto);

}

} using SmartXul.Shared.DTOs;

using SmartXul.Shared.DTOs.Exams;

using SmartXul.Shared.Enums;

namespace SmartXul.Api.Services.Interfaces

{

public interface IExamService

{

// Exam Management

Task<ExamDto> CreateExamAsync(CreateExamDto createExamDto, Guid teacherId);

Task<ExamDto> UpdateExamAsync(Guid examId, UpdateExamDto updateExamDto);

Task<bool> DeleteExamAsync(Guid examId);

Task<ExamDto> GetExamByIdAsync(Guid examId);

Task<IEnumerable<ExamDto>> GetExamsBySchoolAsync(Guid schoolId);

Task<IEnumerable<ExamDto>> GetExamsBySubjectAsync(Guid subjectId, Guid termId);

Task<IEnumerable<ExamDto>> GetExamsByGradeAsync(Guid gradeId, Guid termId);

Task<IEnumerable<ExamDto>> GetUpcomingExamsAsync(Guid schoolId, DateTime fromDate, DateTime toDate);

// Exam Publication

Task<bool> PublishExamAsync(Guid examId);

Task<bool> UnpublishExamAsync(Guid examId);

// Student Registration

Task<bool> RegisterStudentForExamAsync(Guid examId, Guid studentId);

Task<bool> RegisterStudentsForExamAsync(Guid examId, List<Guid> studentIds);

Task<bool> UnregisterStudentFromExamAsync(Guid examId, Guid studentId);

Task<IEnumerable<ExamRegistrationDto>> GetExamRegistrationsAsync(Guid examId);

// Exam Sessions

Task<ExamSessionDto> StartExamSessionAsync(StartExamSessionDto startSessionDto);

Task<ExamSessionDto> GetExamSessionAsync(Guid sessionId);

Task<bool> EndExamSessionAsync(Guid sessionId);

Task<bool> PauseExamSessionAsync(Guid sessionId);

Task<bool> ResumeExamSessionAsync(Guid sessionId);

// Answer Submission

Task<bool> SubmitAnswerAsync(SubmitExamAnswerDto submitAnswerDto);

Task<bool> SaveAnswerDraftAsync(SubmitExamAnswerDto saveAnswerDto);

Task<bool> SubmitExamAsync(Guid sessionId);

// Marking & Results

Task<bool> MarkAnswerAsync(Guid answerId, decimal marks, string feedback);

Task<bool> AutoMarkExamAsync(Guid examId);

Task<ExamResultDto> GenerateExamResultAsync(Guid sessionId);

Task<bool> PublishResultsAsync(Guid examId);

Task<IEnumerable<ExamResultDto>> GetExamResultsAsync(Guid examId);

Task<ExamResultDto> GetStudentExamResultAsync(Guid examId, Guid studentId);

// Statistics & Analytics

Task<ExamStatisticsDto> GetExamStatisticsAsync(Guid examId);

Task<IEnumerable<QuestionAnalysisDto>> GetQuestionAnalysisAsync(Guid examId);

Task<byte[]> GenerateExamReportAsync(Guid examId);

Task<byte[]> GenerateStudentExamResultAsync(Guid examResultId);

// Supervision & Security

Task<bool> AssignSupervisorAsync(Guid examId, Guid teacherId, SupervisorRole role);

Task<bool> RemoveSupervisorAsync(Guid examId, Guid teacherId);

Task<bool> CheckInSupervisorAsync(Guid examId, Guid teacherId);

Task<bool> CheckOutSupervisorAsync(Guid examId, Guid teacherId);

Task<bool> ReportIncidentAsync(CreateExamIncidentDto incidentDto);

Task<IEnumerable<ExamIncidentDto>> GetExamIncidentsAsync(Guid examId);

// Student Experience

Task<IEnumerable<ExamDto>> GetStudentUpcomingExamsAsync(Guid studentId);

Task<IEnumerable<ExamResultDto>> GetStudentExamHistoryAsync(Guid studentId);

Task<ExamDto> GetStudentExamDetailsAsync(Guid examId, Guid studentId);

Task<bool> ValidateExamAccessAsync(Guid examId, Guid studentId);

// Handwriting Integration

Task<bool> ProcessHandwrittenAnswersAsync(Guid examId);

Task<bool> ReviewHandwritingRecognitionAsync(Guid answerId, string correctedText);

}

} namespace SmartXul.Api.Services.Interfaces

{

public interface IFileStorageService

{

Task<string> SaveImageAsync(byte[] imageData, string relativePath);

Task<string> SaveFileAsync(byte[] fileData, string relativePath, string contentType);

Task<byte[]> GetFileAsync(string filePath);

Task<bool> DeleteFileAsync(string filePath);

Task<string> DownloadModelAsync(string cloudPath, Guid modelId);

Task<string> UploadModelAsync(string localPath, string cloudPath);

Task<List<string>> GetFilesInDirectoryAsync(string directoryPath);

Task<bool> FileExistsAsync(string filePath);

Task<long> GetFileSizeAsync(string filePath);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface IGradingService

{

Task<IEnumerable<GradingSchemeDto>> GetAllGradingSchemesAsync();

Task<GradingSchemeDto> CreateGradingSchemeAsync(CreateGradingSchemeDto createDto);

Task<GradingSchemeDto> UpdateGradingSchemeAsync(Guid schemeId, UpdateGradingSchemeDto updateDto);

Task<bool> DeleteGradingSchemeAsync(Guid schemeId);

Task<GradingScaleDto> AddGradingScaleAsync(Guid schemeId, CreateGradingScaleDto createDto);

Task<GradingScaleDto> UpdateGradingScaleAsync(Guid scaleId, UpdateGradingScaleDto updateDto);

Task<bool> DeleteGradingScaleAsync(Guid scaleId);

Task<GradeCalculationDto> CalculateGradeAsync(Guid gradingSchemeId, decimal percentage);

Task<IEnumerable<GradingScaleDto>> GetGradingScalesAsync(Guid schemeId);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface IHandwritingRecognitionService

{

Task<HandwritingRecognitionResultDto> ProcessHandwritingAsync(ProcessHandwritingDto processDto);

Task<TrainingExerciseDto> GenerateTrainingExerciseAsync(GenerateTrainingExerciseDto generateDto);

Task<StudentTrainingSessionDto> StartTrainingSessionAsync(Guid studentId, Guid exerciseId);

Task<StudentTrainingSessionDto> SubmitTrainingDataAsync(SubmitTrainingDataDto submitDto);

Task<bool> VerifyTrainingDataAsync(Guid trainingDataId, string verifiedText);

Task<ModelTrainingJobDto> StartModelTrainingAsync(Guid studentId);

Task<ModelTrainingJobDto> GetTrainingJobStatusAsync(Guid jobId);

Task<HandwritingModelDto> GetStudentModelAsync(Guid studentId);

Task<bool> DeployModelLocallyAsync(Guid modelId);

Task<TrainingProgressDto> GetStudentTrainingProgressAsync(Guid studentId);

Task<IEnumerable<HandwritingTrainingDataDto>> GetUnverifiedTrainingDataAsync(Guid teacherId);

}

} using SmartXul.Api.Models.SupportingModels;

namespace SmartXul.Api.Services.Interfaces

{

public interface IImageProcessingService

{

Task<ImageProcessingResult> ProcessImageAsync(string imagePath);

Task<TextExtractionResult> ExtractTextAsync(string imagePath);

Task<List<CharacterSegment>> SegmentCharactersAsync(string imagePath);

Task<string> PreprocessImageAsync(string imagePath);

Task<bool> ValidateImageQualityAsync(string imagePath);

}

} using SmartXul.Api.Models;

using SmartXul.Api.Models.SupportingModels;

namespace SmartXul.Api.Services.Interfaces

{

public interface IMachineLearningService

{

Task<LocalRecognitionResult> RecognizeTextLocallyAsync(string modelPath, string imagePath);

Task<CloudRecognitionResult> RecognizeTextInCloudAsync(string imagePath);

Task<ModelTrainingResult> TrainPersonalModelAsync(List<HandwritingTrainingData> trainingData);

Task<bool> ValidateModelAsync(string modelPath);

Task<ModelMetrics> EvaluateModelAsync(string modelPath, List<HandwritingTrainingData> testData);

Task<string> DeployModelToCloudAsync(string localModelPath);

Task<bool> UpdateModelAsync(string existingModelPath, List<HandwritingTrainingData> newTrainingData);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface INotificationService

{

Task<NotificationDto> CreateNotificationAsync(CreateNotificationDto createDto);

Task<bool> SendNotificationAsync(Guid notificationId);

Task<bool> SendBulkNotificationAsync(List<Guid> recipientIds, CreateNotificationDto notificationDto);

Task<IEnumerable<NotificationDto>> GetUserNotificationsAsync(Guid userId, string userType);

Task<bool> MarkAsReadAsync(Guid notificationId);

Task<bool> SendWhatsAppMessageAsync(string phoneNumber, string message);

Task<bool> SendEmailNotificationAsync(string email, string subject, string message);

Task<NotificationStatsDto> GetNotificationStatsAsync(Guid schoolId);

}

} using SmartXul.Shared.DTOs;

using SmartXul.Shared.DTOs.Exams;

namespace SmartXul.Api.Services.Interfaces

{

public interface IPdfGenerationService

{

Task<byte[]> GenerateReportCardPdfAsync(StudentReportCardDto reportCard);

Task<byte[]> GenerateClassResultsPdfAsync(ClassResultSummaryDto classSummary);

Task<byte[]> GenerateTimetablePdfAsync(TimetableDto timetable);

Task<byte[]> GenerateAttendanceReportPdfAsync(AttendanceReportDto attendanceReport);

Task<byte[]> GenerateCertificatePdfAsync(StudentDto student, string certificateType);

Task<byte[]> GenerateTranscriptPdfAsync(StudentDto student, List<ResultDto> results);

Task<byte[]> GenerateStudentExamResultAsync(ExamResultDto resultDto);

Task<byte[]> GenerateExamAnalysisReportAsync(ExamStatisticsDto stats);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface IResultService

{

Task<ResultDto> CreateResultAsync(CreateResultDto createResultDto);

Task<ResultDto> UpdateResultAsync(Guid resultId, UpdateResultDto updateResultDto);

Task<IEnumerable<ResultDto>> GetStudentResultsAsync(Guid studentId, Guid termId);

Task<IEnumerable<ResultDto>> GetClassResultsAsync(Guid gradeId, Guid subjectId, Guid termId);

Task<StudentReportCardDto> GenerateReportCardAsync(Guid studentId, Guid termId);

Task<bool> UploadResultsAsync(Guid teacherId, List<CreateResultDto> results);

Task<ClassResultSummaryDto> GetClassResultSummaryAsync(Guid gradeId, Guid termId);

Task<byte[]> PrintTermResultsAsync(Guid studentId, Guid termId);

Task<byte[]> PrintClassResultsAsync(Guid gradeId, Guid termId);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface ISchoolService

{

Task<IEnumerable<SchoolDto>> GetAllSchoolsAsync();

Task<SchoolDto> GetSchoolByIdAsync(Guid schoolId);

Task<SchoolDto> CreateSchoolAsync(CreateSchoolDto createSchoolDto);

Task<SchoolDto> UpdateSchoolAsync(Guid schoolId, UpdateSchoolDto updateSchoolDto);

Task<bool> DeleteSchoolAsync(Guid schoolId);

Task<IEnumerable<SchoolDto>> GetSchoolsByGradingSchemeAsync(Guid gradingSchemeId);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface IStudentService

{

Task<IEnumerable<StudentDto>> GetStudentsBySchoolAsync(Guid schoolId);

Task<IEnumerable<StudentDto>> GetStudentsByGradeAsync(Guid gradeId);

Task<StudentDto> GetStudentByIdAsync(Guid studentId);

Task<StudentDto> CreateStudentAsync(CreateStudentDto createStudentDto);

Task<StudentDto> UpdateStudentAsync(Guid studentId, UpdateStudentDto updateStudentDto);

Task<bool> DeleteStudentAsync(Guid studentId);

Task<bool> AssignStudentToSubjectsAsync(Guid studentId, List<Guid> subjectIds);

Task<IEnumerable<SubjectDto>> GetStudentSubjectsAsync(Guid studentId, Guid schoolYearId);

Task<StudentTimetableDto> GetStudentTimetableAsync(Guid studentId);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface ITeacherService

{

Task<IEnumerable<TeacherDto>> GetTeachersBySchoolAsync(Guid schoolId);

Task<TeacherDto> GetTeacherByIdAsync(Guid teacherId);

Task<TeacherDto> CreateTeacherAsync(CreateTeacherDto createTeacherDto);

Task<TeacherDto> UpdateTeacherAsync(Guid teacherId, UpdateTeacherDto updateTeacherDto);

Task<bool> DeleteTeacherAsync(Guid teacherId);

Task<bool> AssignTeacherToSubjectAsync(AssignTeacherSubjectDto assignDto);

Task<bool> AssignClassTeacherAsync(AssignClassTeacherDto assignDto);

Task<IEnumerable<SubjectDto>> GetTeacherSubjectsAsync(Guid teacherId, Guid schoolYearId);

Task<IEnumerable<GradeDto>> GetTeacherClassesAsync(Guid teacherId, Guid schoolYearId);

}

} using SmartXul.Shared.DTOs;

namespace SmartXul.Api.Services.Interfaces

{

public interface ITimetableService

{

Task<TimetableDto> GenerateTimetableAsync(GenerateTimetableDto generateDto);

Task<TimetableDto> GetGradeTimetableAsync(Guid gradeId, Guid termId);

Task<StudentTimetableDto> GetStudentTimetableAsync(Guid studentId);

Task<TeacherTimetableDto> GetTeacherTimetableAsync(Guid teacherId, Guid termId);

Task<TimetableSlotDto> UpdateTimetableSlotAsync(Guid slotId, UpdateTimetableSlotDto updateDto);

Task<bool> ValidateTimetableAsync(Guid timetableId);

Task<TimetableConflictDto> CheckTimetableConflictsAsync(Guid timetableId);

Task<byte[]> PrintTimetableAsync(Guid timetableId);

}

} using SmartXul.Api.Models.SupportingModels;

namespace SmartXul.Api.Services.Interfaces

{

public interface IWhatsAppService

{

Task<bool> SendMessageAsync(string phoneNumber, string message);

Task<bool> SendMediaMessageAsync(string phoneNumber, string mediaUrl, string caption);

Task<bool> SendTemplateMessageAsync(string phoneNumber, string templateName, Dictionary<string, string> parameters);

Task<List<WhatsAppMessageStatus>> GetMessageStatusAsync(List<string> messageIds);

}

}