EduPlatform — Regenerated Core Files (Unified model)

> This document contains copy-paste-ready code for Entities (with Fluent API configs), ApplicationDbContext, Repositories, Services, Controllers, simple Razor views, helpers and EF commands — aligned with the **unified Course model** and your requirements.

How to use

1. Copy each file block into the appropriate project/folder in your solution (`Edu.Domain`, `Edu.Infrastructure`, `Edu.Application`, `Edu.Web`).
2. Register all repository/service DI in `Program.cs` (I include a suggested registration snippet near the bottom).
3. Run migrations (commands provided at the end).

> **Note:** Per your instruction I used Fluent API only for EF configuration (no DataAnnotations). I kept controllers and services thin and aligned with Clean Architecture (interfaces in Application, implementations in Infrastructure, MVC in Web).

---

1. Entities (Edu.Domain/Entities)

`CourseType.cs`

```
namespace Edu.Domain.Entities;

public enum CourseType { School = 0, Private = 1 }
```

`Department.cs`

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public class Department
```

```
{
    public int Id { get; set; }
    public string Name { get; set; } = string.Empty;
    public string? Description { get; set; }
    public ICollection<Level> Levels { get; set; } = new List<Level>();
}

public class DepartmentConfiguration : IEntityTypeConfiguration<Department>
{
    public void Configure(EntityTypeBuilder<Department> b)
    {
        b.ToTable("Departments");
        b.HasKey(x => x.Id);
        b.Property(x => x.Name).HasMaxLength(200).IsRequired();
    }
}
```

`Level.cs`

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public class Level
{
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public int DepartmentId { get; set; }
    public Department? Department { get; set; }
    public ICollection<Subject> Subjects { get; set; } = new List<Subject>();
}

public class LevelConfiguration : IEntityTypeConfiguration<Level>
{
    public void Configure(EntityTypeBuilder<Level> b)
    {
        b.ToTable("Levels");
        b.HasKey(x => x.Id);
        b.Property(x => x.Title).HasMaxLength(200).IsRequired();
        b.HasOne(x => x.Department).WithMany(d => d.Levels).HasForeignKey(x =>
x.DepartmentId).OnDelete(DeleteBehavior.Cascade);
    }
}
```

`Subject.cs`

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public class Subject
{
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public string? Description { get; set; }
    public int LevelId { get; set; }
    public Level? Level { get; set; }
    public ICollection<Lesson> Lessons { get; set; } = new List<Lesson>();
    public ICollection<Course> Courses { get; set; } = new List<Course>();
}

public class SubjectConfiguration : IEntityTypeConfiguration<Subject>
{
    public void Configure(EntityTypeBuilder<Subject> b)
    {
        b.ToTable("Subjects");
        b.HasKey(x => x.Id);
        b.Property(x => x.Title).HasMaxLength(250).IsRequired();
        b.HasOne(x => x.Level).WithMany(l => l.Subjects).HasForeignKey(x =>
x.LevelId).OnDelete(DeleteBehavior.Cascade);
    }
}
```

`Course.cs` **(Unified)**

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public class Course
{
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public string? ShortDescription { get; set; }
```

```
    public string? Description { get; set; }
    public int? SubjectId { get; set; }      // optional: link to school subject
    public Subject? Subject { get; set; }
    public string? TeacherId { get; set; }   // ApplicationUser.Id
    public CourseType Type { get; set; } = CourseType.School;
    public bool RequiresApprovalOnEnroll { get; set; } = true;
    public decimal? Price { get; set; }
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
    public ICollection<Lesson> Lessons { get; set; } = new List<Lesson>();
    public ICollection<CourseDocument> Documents { get; set; } = new
List<CourseDocument>();
}

public class CourseConfiguration : IEntityTypeConfiguration<Course>
{
    public void Configure(EntityTypeBuilder<Course> b)
    {
        b.ToTable("Courses");
        b.HasKey(x => x.Id);
        b.Property(x => x.Title).HasMaxLength(300).IsRequired();
        b.Property(x => x.ShortDescription).HasMaxLength(1000);
        b.Property(x => x.Price).HasColumnType("decimal(10,2)");
        b.Property(x => x.Type).HasConversion<int>().IsRequired();
        b.HasOne(x => x.Subject).WithMany(s => s.Courses).HasForeignKey(x =>
x.SubjectId).OnDelete(DeleteBehavior.SetNull);
        b.HasIndex(x => new { x.SubjectId });
    }
}
```

`Lesson.cs`

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public enum LessonType { Video = 0, Document = 1, Text = 2 }

public class Lesson
{
    public int Id { get; set; }
    public int? CourseId { get; set; }
    public Course? Course { get; set; }
    public int? SubjectId { get; set; }
```

```
    public Subject? Subject { get; set; }
    public string Title { get; set; } = string.Empty;
    public string? Content { get; set; }
    public LessonType Type { get; set; } = LessonType.Video;
    public string? VideoProvider { get; set; }
    public string? VideoId { get; set; }
    public string? DocumentUrl { get; set; }
    public int Order { get; set; } = 0;
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
}

public class LessonConfiguration : IEntityTypeConfiguration<Lesson>
{
    public void Configure(EntityTypeBuilder<Lesson> b)
    {
        b.ToTable("Lessons");
        b.HasKey(x => x.Id);
        b.Property(x => x.Title).HasMaxLength(400).IsRequired();
        b.Property(x => x.Type).HasConversion<int>().IsRequired();
        b.HasOne(x => x.Course).WithMany(c => c.Lessons).HasForeignKey(x =>
x.CourseId).OnDelete(DeleteBehavior.Cascade);
        b.HasOne(x => x.Subject).WithMany(s => s.Lessons).HasForeignKey(x =>
x.SubjectId).OnDelete(DeleteBehavior.Cascade);
        b.HasIndex(x => new { x.CourseId, x.SubjectId, x.Order });
    }
}
```

`CourseDocument.cs`

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public class CourseDocument
{
    public int Id { get; set; }
    public int CourseId { get; set; }
    public Course? Course { get; set; }
    public string FileUrl { get; set; } = string.Empty;
    public string? Title { get; set; }
    public DateTime UploadedAt { get; set; } = DateTime.UtcNow;
}
```

```
public class CourseDocumentConfiguration :
IEntityTypeConfiguration<CourseDocument>
{
    public void Configure(EntityTypeBuilder<CourseDocument> b)
    {
        b.ToTable("CourseDocuments");
        b.HasKey(x => x.Id);
        b.Property(x => x.FileUrl).HasMaxLength(500).IsRequired();
        b.HasOne(x => x.Course).WithMany(c => c.Documents).HasForeignKey(x =>
x.CourseId).OnDelete(DeleteBehavior.Cascade);
    }
}
```

`Enrollment.cs`

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public enum EnrollmentStatus { Pending = 0, Enrolled = 1, Cancelled = 2,
Completed = 3 }

public class Enrollment
{
    public int Id { get; set; }
    public int CourseId { get; set; }
    public Course? Course { get; set; }
    public string StudentId { get; set; } = string.Empty; // ApplicationUser.Id
    public EnrollmentStatus Status { get; set; } = EnrollmentStatus.Pending;
    public DateTime RequestedAt { get; set; } = DateTime.UtcNow;
    public DateTime? EnrolledAt { get; set; }
    public string? AdminNote { get; set; }
}

public class EnrollmentConfiguration : IEntityTypeConfiguration<Enrollment>
{
    public void Configure(EntityTypeBuilder<Enrollment> b)
    {
        b.ToTable("Enrollments");
        b.HasKey(x => x.Id);
        b.HasIndex(x => new { x.CourseId, x.StudentId }).IsUnique();
        b.Property(x => x.Status).HasConversion<int>();
```

```
    }
}
```

`Booking.cs` **(simple)**

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;

namespace Edu.Domain.Entities;

public enum BookingStatus { Pending = 0, Confirmed = 1, Cancelled = 2,
Completed = 3 }

public class Booking
{
    public int Id { get; set; }
    public int CourseId { get; set; }
    public string StudentId { get; set; } = string.Empty;
    public string TeacherId { get; set; } = string.Empty;
    public DateTime SlotStart { get; set; }
    public DateTime SlotEnd { get; set; }
    public BookingStatus Status { get; set; } = BookingStatus.Pending;
    public string? Note { get; set; }
}

public class BookingConfiguration : IEntityTypeConfiguration<Booking>
{
    public void Configure(EntityTypeBuilder<Booking> b)
    {
        b.ToTable("Bookings");
        b.HasKey(x => x.Id);
        b.Property(x => x.Status).HasConversion<int>();
    }
}
```

`Assignment.cs` **&** `AssignmentSubmission.cs`

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
```

```
namespace Edu.Domain.Entities;

public class Assignment
{
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public string? Description { get; set; }
    public int CourseId { get; set; }
    public Course? Course { get; set; }
    public int? LessonId { get; set; }
    public Lesson? Lesson { get; set; }
    public DateTime DueDate { get; set; }
    public int MaxPoints { get; set; } = 100;
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
    public ICollection<AssignmentSubmission> Submissions { get; set; } = new
List<AssignmentSubmission>();
}

public class AssignmentSubmission
{
    public int Id { get; set; }
    public int AssignmentId { get; set; }
    public Assignment? Assignment { get; set; }
    public string StudentId { get; set; } = string.Empty;
    public string? FileUrl { get; set; }
    public string? TextAnswer { get; set; }
    public DateTime SubmittedAt { get; set; } = Da
```