# Education portal plans

## 1. Technical Overview

### *1.1 Technologies and Tooling*

- **ASP.NET Core 9 MVC**
  Used as the main presentation layer of the application. Provides controllers, views, and routing to implement the user interface.
- **Entity Framework Core 9 (Code First, Fluent API)**
  Used as the Object-Relational Mapper (ORM) for data access. Fluent API configurations define relationships, keys and TPT inheritance mappings.
- **Microsoft SQL Server**
  Relational database management system to persist all application data.
- **ASP.NET Core Identity**
  Provides authentication and authorization features. Handles user registration, login, password management, and role-based access.
- **C# 12**
  Main programming language used for business logic, data access, and presentation layer code.
- **xUnit + Moq**
  Unit testing framework with mocked repositories used to validate business/services logic.
- **IDE**: Visual Studio 2022 (latest)
- **CLI**: dotnet (build, test, EF migrations)

### *1.2 Architecture*

**Layered (N-tier) architecture:**

- **Presentation Layer (UI)** – ASP.NET Core MVC project containing controllers, view models, and Razor views
- **Business Logic Layer (BLL)** – Application services, business rules, DTOs/mappers, validation.
- **Data Access Layer (DAL)** – Entities, Enums, EF Core DbContext, Fluent API configurations, Migrations, TPT mappings, repository interfaces and implementations, Unit of Work, ASP.NET Core Identity EF stores

## *1.3 Design Patterns*

- **Repository Pattern**
  Encapsulates data access logic, provides interfaces for querying and persisting entities.
- **Unit of Work (via DbContext)**
  Tracks changes and saves them as a single transaction.
- **DTOs (Data Transfer Objects)**
  Used to transfer data between layers, avoiding direct exposure of entities to the presentation layer.
- **Dependency Injection (DI)**
  Built-in ASP.NET Core DI container is used to register and inject services and repositories.

## *1.4 Inheritance Mapping*

- **Strategy:** TPT (Table per Type) for the Material hierarchy (Material base + VideoMaterial, BookMaterial, ArticleMaterial derived types).

## *1.5 Identity Integration*

- **ApplicationUser** extends IdentityUser<Guid> with additional profile fields (FullName, CreatedAt).
- Other entities reference ApplicationUser.Id via foreign keys.
- ASP.NET Core Identity provides authentication, registration, and authorization.
- **Layer placement:**
  - **Data Access Layer:** Identity EF stores over AppDbContext
  - **Presentation:** Identity UI (scaffolded pages for Login/Register/Logout)
  - **Business Logic:** no direct Identity dependency; services receive UserId or use an IUserContext abstraction to get the current user id.
- **Authorization rules:**
  - [Authorize] on create/edit course/materials/skills, enroll (start a course), mark course/material complete, and profile operations.
  - Only logged in user can create or edit
- **Registration settings:** For registration, user needs to add an email address and a password. options.User.RequireUniqueEmail = true; demo-friendly password for now.

## *1.6 Testing* (Unit tests)

- **Tools:** xUnit + Moq (mocked repositories, no real database)
- **Scope:** unit tests for Application (no integration/UI tests)
- **Approach:**
  - Arrange – Act – Assert; one service per test class (e.g., EnrollmentServiceTests)
  - Mock ICourseRepository, IMaterialRepository, IUserCourseRepository, IUserMaterialRepository, ISkillRepository
  - Verify interactions (e.g., AddAsync/UpdateAsync called once, SaveChangesAsync called once)
- **Goals:** validate critical business rules quickly and reliably:
  - Enrollment set initial status (InProgress) / progress (0 %) correctly
  - Marking a material as complete updates progress (100% means all materials are completed) across all courses sharing that material
  - Auto-complete course at 100% and award skills (+1 if already owned)
  - Attach/detach materials does not create duplicates and returns materials sorted as specified

## *1.7 Naming Conventions*

- **Entities (C# classes):** singular – Course, Material, Skill, ApplicationUser
- **Database tables / DbSet names:** plural – Courses, Materials, Skills, UserCourses, CourseMaterials, UserMaterials
  TPT tables: Materials, VideoMaterials, BookMaterials, ArticleMaterials
- **Columns / keys / FKs:** singular property names – Id, CourseId, MaterialId, UserId, CreatedAt
- **DbContext sets:** DbSet<Course> Courses, DbSet<Material> Materials, etc.
- **Repositories / Services:** singular type names; interfaces prefixed with I – ICourseRepository → CourseRepository, ICourseService → CourseService.
  All asynchronous operations use the *Async suffix – GetByIdAsync, AddAsync, UpdateAsync, SaveChangesAsync.
- **Unit of Work:** IUnitOfWork with SaveChangesAsync().
- **Controllers / routes / view folders:** plural – CoursesController (/Courses), MaterialsController (/Materials), SkillsController, EnrollmentsController, ProfileController. Razor view folders mirror controller names (e.g., Views/Courses/*).

- **ViewModels / DTOs / form models:** singular – CourseFormViewModel, MaterialDto, SkillDto.
- **Partial views:** underscore prefix – _CourseForm.cshtml, _MaterialForm.cshtml, _SkillForm.cshtml, _CourseCard.cshtml, _ModalShell.cshtml.
- **Test classes:** <TypeUnderTest>Tests – CourseServiceTests, EnrollmentServiceTests. Test project names reflect layer – EducationPortal.BusinessLogic.UnitTests.
- **Namespaces:** layer-scoped – EducationPortal.DataAccess, EducationPortal.BusinessLogic, EducationPortal.Presentation.
- **C# casing:** PascalCase for types/methods/properties; camelCase for locals/parameters; enums are singular types with PascalCase members.

# 2. Entities, relations

**Nullability convention:** Unless explicitly marked with **(NULL)**, all columns are **NOT NULL** (required).

## *2.1 AspNetUsers* (table in ASP.NET Identity)

### ApplicationUser

- **Id**: uniqueidentifier (PK) – Guid
- **UserName**, **NormalizedUserName**, **Email** (default Identity fields)
- **FullName**: nvarchar(128) (NULL)
- **CreatedAt**: datetime2 (UTC)

### Relations:

- ApplicationUser – Skill: **N–N** (via **UserSkill**)
- ApplicationUser – Course: **N–N** (via **UserCourse**)
- ApplicationUser – Material: **N–N** (via **UserMaterial**)

## *2.2 Skill*

- **Id**: int (PK, identity)
- **Name**: nvarchar(100) (Unique)
- **Description**: nvarchar(500) (NULL)

### Relations:

- Skill – ApplicationUser: **N–N** (via **UserSkill**)

- Skill – Course: **N–N** (via **CourseSkill**)

### *2.3 UserSkill* (junction with level)

- **UserId**: uniqueidentifier (PK1, FK → AspNetUsers.Id)
- **SkillId**: int (PK2, FK → Skills.Id)
- **Level**: int (>=1)

**Relations:**

- UserSkill → ApplicationUser: **N–1**
- UserSkill → Skill: **N–1**

### *2.4 Course*

- **Id:** int (PK, identity)
- **Name:** nvarchar(200)
- **Description:** nvarchar(1000) (NULL)

**Relations:**

- Course – Skill: **N–N** (via **CourseSkill**)
- Course – Material: **N–N** (via **CourseMaterial**)
- Course – ApplicationUser (enrollments): **N–N** (via **UserCourse**)

### *2.5 CourseSkill* (skills awarded on completion)

- **CourseId:** int (PK1, FK → Courses.Id)
- **SkillId:** int (PK2, FK → Skills.Id)

**Relations:**

- CourseSkill → Course: **N–1**
- CourseSkill → Skill: **N–1**

### *2.6 Material* (TPT: Material (base) + VideoMaterial, BookMaterial, ArticleMaterial derived types)

**Material (base):**

- **Id:** int (PK, identity)

- **Title:** nvarchar(200)
- **Description:** nvarchar(1000) (NULL)

**VideoMaterial:**

- **Id:** int (PK, FK → Materials.Id)
- **DurationSec:** int
- **HeightPx:** int
- **WidthPx:** int

**BookMaterial:**

- **Id:** int (PK, FK → Materials.Id)
- **Authors:** nvarchar(300)
- **Pages:** int
- **FormatId:** int (FK → BookFormats.Id)
- **PublicationYear:** int

**ArticleMaterial:**

- **Id:** int (PK, FK → Materials.Id)
- **PublishedAt:** date
- **SourceUrl:** nvarchar(500)

**Relations:**

- Material – Course: **N–N (**via **CourseMaterial)**
- Material – ApplicationUser: **N–N (**via **UserMaterial)**

## *2.7 CourseMaterial* (junction)

- **CourseId:** int (PK1, FK → Courses.Id)
- **MaterialId:** int (PK2, FK → Materials.Id)

**Relations:**

- CourseMaterial → Course: **N–1**
- CourseMaterial → Material: **N–1**

Materials may exist without being attached to any course (0..N).

### *2.8 UserCourse* (junction with enrollments, status, progress)

- **UserId:** uniqueidentifier (PK1, FK → AspNetUsers.Id)
- **CourseId:** int (PK2, FK → Courses.Id)
- **StatusId:** int (FK → CourseStatuses.Id)
- **ProgressPercent:** tinyint (0-100)

**Relations:**

- UserCourse → ApplicationUser: **N–1**
- UserCourse → Course: **N–1**

### *2.9 UserMaterial* (completed materials, junction)

- **UserId:** uniqueidentifier (PK1, FK → AspNetUsers.Id)
- **MaterialId:** int (PK2, FK → Materials.Id)

**Relations:**

- UserMaterial → ApplicationUser: **N–1**
- UserMaterial → Material: **N–1**

## 2.10 Lookup tables

**BookFormats**

- **Id:** int (PK) (values start from 1)
- **Name:** nvarchar(50) (e.g., PDF, EPUB, MOBI)

**CourseStatuses**

- **Id:** int (PK) (values start from 1)
- **Name:** nvarchar(50) (e.g., InProgress, Completed)

### *2.11 Relations* (overview)

- **ApplicationUser – Skill: N–N** via **UserSkill**
- **ApplicationUser – Course** (enrollment)**: N–N** via **UserCourse**
- **ApplicationUser – Material** (progress)**: N–N** via **UserMaterial**
- **Course – Skill: N–N** via **CourseSkill**

- **Course – Material: N–N** via **CourseMaterial**
- **CourseSkill:** links **Course (N–1)** to **Skill (N–1)**
- **CourseMaterial:** links **Course (N–1)** to **Material (N–1)**
- **UserCourse:** links **ApplicationUser (N–1)** to **Course (N–1)**
- **UserMaterial:** links **ApplicationUser (N–1)** to **Material (N–1)**

# 3. Project Structure (layered + repo pattern + MVC)

## *3.1 Solution layout*

- **EducationPortal.DataAccess (DAL)**
  Entities, EF Core AppDbContext, Fluent API (incl. TPT), Migrations, repository implementations, UnitOfWork implementation, Identity EF stores

- **EducationPortal.BusinessLogic (BLL)**
  Application services, business rules, DTOs/mappers, repository interfaces (e.g., ICourseRepository), IUnitOfWork, IUserContext

- **EducationPortal.Presentation (UI)**
  ASP.NET Core MVC controllers, view models, Razor views/partials, filters, DI composition (Program.cs), Identity UI integration

- **EducationPortal.BusinessLogic.UnitTests**
  xUnit + Moq tests for BLL services (mocked repositories + IUnitOfWork)

## *3.2 Service (Logic) Layer*

**Services:**

- **CourseService**
  - Create/Edit course (name, description, skills)
  - Attach/Detach materials
  - Browse, Details
- **MaterialService**
  - Create/Edit/Attach/Detach materials (handles TPT specifics)
  - List by type
- **EnrollmentService**
  - Enroll / MarkMaterialComplete / CompleteCourse

    - Recalculate progress; propagate completion to other courses sharing the same material
    - Course progress = (completed materials / total materials) × 100. At 100% the course is auto-completed and skills are awarded
- **SkillService**
    - Create/Edit/Attach/Detach Skills
    - Grant/Increment skills on course completion; list user skills
- **ProfileService**
    - “My Courses” (Available / InProgress / Completed) with progress %
    - “My Skills” (with levels)

**Authentication/Authorization:** via ASP.NET Core Identity (no custom **AuthService** needed).

# 4. Minimal Feature Scope

## *4.1 MVP (3-week deliverable)*

- Registration/Login (Identity)
- Courses list + details
- Create/Edit Course (name, description, add materials from list or create new, map skills)
- Enroll to course
- Mark a material as Complete → update progress across all courses that contain the same material
- Complete course → award skills (+1 level if already owned)
- Profile: personal info, My Skills (levels), My Courses: Available / In Progress / Completed (+ progress %)

## *4.2 Nice-to-have (time permitting)*

- Search/filters (by skill, type, keyword)
- Course cover image upload
- Audit fields (CreatedBy/UpdatedBy + timestamps)
- Database indexes

- **CourseMaterials (MaterialId, CourseId)** – speeds up progress propagation (material → affected courses)
- **UserCourses (UserId, Status)** – fast listing for "My Courses" tabs
- **Materials.Title** – for title-based search/filtering

# 5. MVC (Controllers and Views)

## *5.1 Controllers*

- **CoursesController**
  - **GET /Courses → Index** (browse all courses; cards show **Details** only)
  - **GET /Courses/Details/{id} → Details**

    Shows materials in the course (with "Mark complete" if logged in), and skills awarded on completion.

    **Buttons on this page:**

    - **Edit course** → opens Course form modal (prefilled)
    - **Add new material to this course** → opens Material create modal (partial) with this course preselected
    - **Add new skill to this course** → opens Skill create modal (partial) with this course preselected
  - **GET /Courses/Create** → returns Views/Courses/Partials/_CourseForm.cshtml (modal)
  - **POST /Courses/Create** → create course
    **Input:** CourseFormViewModel + SelectedMaterialIds[], SelectedSkillIds[]
  - **GET /Courses/Edit/{id}** → returns _CourseForm (modal, prefilled)
  - **POST /Courses/Edit/{id}** → update course
    **Input:** CourseFormViewModel + SelectedMaterialIds[], SelectedSkillIds[]

**Pickers inside the Course form modal (existing items only):**

- **GET /Courses/MaterialPickerList** → returns partial with a checkbox list of existing materials
- **GET /Courses/SkillPickerList** → returns partial with a checkbox list of existing skills

- **MaterialsController**
  - **GET /Materials → Index** (filter by type)
  - **GET /Materials/Details/{id} → Details**
  - **GET /Materials/Create** → returns Views/Materials/Partials/_MaterialForm.cshtml (modal)
  - **POST /Materials/Create** → create material
    **Input:** MaterialFormViewModel + SelectedCourseIds[]
  - **GET /Materials/Edit/{id}** → returns _MaterialForm (modal, prefilled)
  - **POST /Materials/Edit/{id}** → edit material
    **Input:** MaterialFormViewModel + SelectedCourseIds[]

- **SkillsController**
  - **GET /Skills → Index**
  - **GET /Skills/Details/{id} → Details**
  - **GET /Skills/Create** → returns Views/Skills/Partials/_SkillForm.cshtml (modal)
  - **POST /Skills/Create** → create skill
    **Input:** SkillFormViewModel + SelectedCourseIds[]
  - **GET /Skills/Edit/{id}** → returns _SkillForm (modal, prefilled)
  - **POST /Skills/Edit/{id}** → edit skill
    **Input:** SkillFormViewModel + SelectedCourseIds[]

- **EnrollmentsController**
  - **POST /Enrollments/Enroll/{courseId}** → enroll (InProgress, 0%).
  - **POST /Enrollments/MarkMaterialComplete/{materialId}** → mark complete; propagate to all courses containing that material; auto-complete course at 100%.
  - **POST /Enrollments/CompleteCourse/{courseId}** → idempotent finalize + award skills (+1 if already owned).

- **ProfileController**
    - **GET /Profile/MyProfile**
    - **GET /Profile/MyCourses** → tabs: **Available / InProgress / Completed** (with progress %)
    - **GET /Profile/MySkills** → user skills with levels

- **Account**
    - **Login/Register/Logout** via Identity UI (Areas/Identity/Pages/Account/*).

## *5.2 Views (Razor)*

- **Courses**
    - **Views/Courses/Index.cshtml** – grid of Shared/Partials/_CourseCard.cshtml (Details button only)
    - **Views/Courses/Details.cshtml**
        - **Sections:**
            - Materials (with "Mark complete" buttons if logged in)
            - Skills awarded on completion
        - **Buttons (top-right):** Edit course, Add new material to this course, Add new skill to this course
    - **Views/Courses/Partials/_CourseForm.cshtml** – modal with:
        - Basic fields (Name, Description)
        - **Materials picker** (checkbox list of existing materials; list from /Courses/MaterialPickerList)
        - **Skills picker** (checkbox list of existing skills; list from /Courses/SkillPickerList)
        - Save/Cancel

- **Materials**
    - **Views/Materials/Index.cshtml**
    - **Views/Materials/Details.cshtml**
    - **Views/Materials/Partials/_MaterialForm.cshtml** – modal with:

        - Base + derived fields (Video/Book/Article) shown conditionally
        - **Course picker**

- **Skills**
    - **Views/Skills/Index.cshtml**
    - **Views/Skills/Partials/_SkillForm.cshtml** – modal **with** Course picker (SelectedCourseIds[])

- **Profile**
    - **Views/Profile/MyCourses.cshtml** – tabs + progress badges
    - **Views/Profile/MySkills.cshtml**
    - **Views/Profile/MyProfile.cshtml**

- **Account (Identity UI)**
    - **Areas/Identity/Pages/Account/Login.cshtml**
    - **Areas/Identity/Pages/Account/Register.cshtml**
    - **Areas/Identity/Pages/Account/Logout.cshtml**

- **Shared**
    - **Views/Shared/_Layout.cshtml**
    - **Views/Shared/Modals/_ModalShell.cshtml**
    - **Views/Shared/Partials/_CourseCard.cshtml**
    - **Views/Shared/Partials/_ValidationSummary.cshtml**

# 6. Timeline

## *6.1 Planning*

- Finalize entities and Fluent API design
- Confirm relations, enums, validation rules
- Produce DB schema diagram (ERD)
- Wireframes for main screens (Courses Index/Details, Materials Index/Details, Skills Index, Profile pages)

- Align surface of controllers/services
- Document Identity settings (RequireUniqueEmail, demo-friendly password) and authorization rules

## *6.2 Week 1 – Heavy Build*

### Main task 1: Solution skeleton and infrastructure

- Create solution with 4 projects: EducationPortal.DataAccess, EducationPortal.BusinessLogic, EducationPortal.Presentation, EducationPortal.BusinessLogic.UnitTests
- Add NuGets (EF Core 9 + SqlServer + Design, Identity, xUnit, Moq)
- Wire DI in Presentation: register repos + IUnitOfWork, Application services, and IUserContext

### Main task 2: Data modeling

- Implement entities/enums
- AppDbContext Fluent API configs (including TPT mappings)
- Configure junctions (CourseMaterial, UserCourse, UserMaterial) with composite PKs/FKs
- Integrate Identity EF stores in the same DbContext
- Create initial migration + database

### Main task 3: Repository + UoW

- Define repository interfaces (ICourseRepository, IMaterialRepository, ISkillRepository, IUserCourseRepository, IUserMaterialRepository, IUserRepository) + IUnitOfWork
- Implement EF-based repositories + UoW
- Register with DI

## *6.3 Week 2 – Heavy Build*

### Main task 1: Core services and CRUD UI

- Implement CourseService, MaterialService, SkillService
- Controllers:
    - CoursesController (Index, Details, Create/Edit modal with pickers)
    - MaterialsController (Index/Details/Create/Edit modal)
    - SkillsController (Index/Create/Edit modal)
- Views/Partials: modal shell, _CourseForm (with Material/Skill pickers of existing items), _MaterialForm (type-specific sections), _SkillForm, _CourseCard, layout
- Identity UI for register / login

### Main task 2: Seed and smoke tests

- Seed 3–4 skills, 6–8 materials (Video/Book/Article), 2–3 demo courses (overlapping materials)
- Manual smoke: create course, attach existing materials/skills, edit course; open Materials/Skills CRUD modals

### Main task 3: Enrollment and progress engine

- EnrollmentService: Enroll, MarkMaterialComplete, CompleteCourse (idempotent).
- Recalculate per-course progress; propagate material completion to all courses containing it.
- At 100%: auto-complete course + award skills (+1 if already owned)

### Main task 4: Profile area

- ProfileService + ProfileController: MyCourses (Available / InProgress / Completed with %), MySkills, MyProfile

## 6.4 Week 3 – Authorization, Validation, Polish, QA

### Main task 1: Authorization and validation

- [Authorize] on mutating endpoints
- Enforce “only logged in user can create / edit course” in Application layer
- Server-side validation for forms (required, ranges, enums)

### Main task 2: Unit tests (xUnit + Moq)

- EnrollmentService: propagation, idempotency, skill award
- CourseService: attach/detach prevents duplicates; materials returned in expected order
- Optional: SkillService increments

### Main task 3: UX polish

- Partial views tidy-up; validation summaries/messages
- On Course Details page: buttons Edit course, Add new material to this course, Add new skill to this course (open create modals with current course preselected)

### Main task 4: QA and bug triage

- Full manual test pass (happy paths + key edge cases); fix critical/major issues

### Main task 5: Docs and demo readiness

- README.md (setup/run/tech overview), architecture diagram, ERD checked in
- Seed data finalized for demo

# 7. ERD diagram