**Project Documentation – UniversityApp**

**Alex – Zein**

**Github link : https://github.com/alexjuv1/eece437\_ass1\_alexandzein.git**

**Application Structure**

The project follows **Clean Architecture** and includes four distinct layers:

**1. Core Layer (UniversityApp.Core)**

Contains the domain entities and repository interfaces:

* Entities/Student.cs
* Entities/Professor.cs
* Entities/Course.cs
* Entities/Department.cs
* Interfaces/IStudentRepository.cs
* Interfaces/IProfessorRepository.cs
* Interfaces/ICourseRepository.cs
* Interfaces/IDepartmentRepository.cs

**2. Application Layer (UniversityApp.Application)**

Implements the business logic using:

* **DTOs**:
  + DTOs/StudentDto.cs, CreateStudentDto.cs, UpdateStudentDto.cs
  + DTOs/ProfessorDto.cs, CreateProfessorDto.cs, UpdateProfessorDto.cs
  + DTOs/CourseDto.cs, CreateCourseDto.cs, UpdateCourseDto.cs
  + DTOs/DepartmentDto.cs, CreateDepartmentDto.cs, UpdateDepartmentDto.cs
* **CQRS Handlers (using MediatR)**:
  + Features/Students/Commands/... and Features/Students/Queries/...
  + Features/Professors/Commands/... and Features/Professors/Queries/...
  + Features/Courses/Commands/... and Features/Courses/Queries/...
  + Features/Departments/Commands/... and Features/Departments/Queries/...
* **Validation** (using FluentValidation):
  + Validators/CreateStudentDtoValidator.cs, UpdateStudentDtoValidator.cs
  + Validators/CreateProfessorDtoValidator.cs, UpdateProfessorDtoValidator.cs
  + Validators/CreateCourseDtoValidator.cs, UpdateCourseDtoValidator.cs
  + Validators/CreateDepartmentDtoValidator.cs, UpdateDepartmentDtoValidator.cs
* **Common Utilities**:
  + Common/Result.cs (Result wrapper for responses)
* **Mappings** (AutoMapper profiles):
  + Mappings/StudentProfile.cs
  + Mappings/ProfessorProfile.cs
  + Mappings/CourseProfile.cs
  + Mappings/DepartmentProfile.cs

**3. Infrastructure Layer (UniversityApp.Infrastructure)**

Handles data access:

* Persistence/AppDbContext.cs (configured for Identity + EF Core)
* Persistence/Repositories/StudentRepository.cs
* Persistence/Repositories/ProfessorRepository.cs
* Persistence/Repositories/CourseRepository.cs
* Persistence/Repositories/DepartmentRepository.cs

**4. Web Layer (UniversityApp.Web)**

ASP.NET Core app with:

* API Controllers:
  + Controllers/StudentsController.cs
  + Controllers/ProfessorsController.cs
  + Controllers/CoursesController.cs
  + Controllers/DepartmentsController.cs
* Program and Middleware setup:
  + Program.cs (MediatR, AutoMapper, FluentValidation, EF setup, JWT Auth, Swagger)
  + appsettings.json (JWT secrets, connection string)
  + Properties/launchSettings.json

**Implemented Features (User Stories)**

**Student Management**

* Admin can create, update, delete, and view students.
* Students can view their profile.

**Professor Management**

* Admin can manage professors, including assigning departments and emails.

**Course Management**

* Admin can assign and manage courses and their credits.

**Department Management**

* Admin can create, edit, delete, and assign departments to professors.

**Pagination Support**

* All GET list endpoints support pagination via:  
  ?pageNumber=1&pageSize=10

**Authentication (JWT-Based)**

* Login implemented with JWT token generation.
* Users are authenticated and authorized using [Authorize] attribute.
* Roles: Admin, Student.
* Seeded default users with role-based access.

**Key Design Decisions**

1. Used **Clean Architecture** for clear separation of concerns and testability.
2. **MediatR** for implementing CQRS and decoupling logic from controllers.
3. **FluentValidation** for clear and reusable DTO validation rules.
4. **AutoMapper** for mapping between DTOs and domain models.
5. Used **EF Core with SQLite** for lightweight database storage suitable for assignments.
6. **JWT Authentication** to secure APIs and enable role-based access control.

**Setup Instructions**

1. **Restore dependencies**:

dotnet restore

1. **Apply EF Core migrations**:

dotnet ef database update --project UniversityApp.Infrastructure

1. **Run the web project**:

dotnet run --project UniversityApp.Web

1. **Visit Swagger API Docs**:

bash

CopyEdit

https://localhost:{PORT}/swagger

**Default Users (automatically seeded)**

* Admin:  
  Email: admin@uni.com  
  Password: Admin1!
* Student:  
  Email: stud@uni.com  
  Password: Stud1!

UML :

