# ASP.NET Advanced – June 2025

This document defines the **requirements for the** **final project assignment**   
for the ["ASP.NET Advanced" Course @ SoftUni](https://softuni.bg/trainings/4954/asp-net-advanced-june-2025).

# Individual Project Assignment 17 August 2025

## General Requirements

Your Web application should use the following technologies, frameworks, and development techniques:

* The application must be implemented using **ASP.NET Core** Framework - **.NET 6.0** or **.NET 8.0**
  + The application must have at least **15** web pages (**views**)
  + The application must have at least **6** **entity models**
  + The application must have at least **6** **controllers**
* Use **Visual Studio 2022 / JetBrains Project Rider**
  + Use the **Razor** template engine for generating the UI
    - Use **sections** and **partial views**
  + Optionally, you could also use Web API to create a RESTful service and use JavaScript / TypeScript for the **Front-End**
* Use **Microsoft SQL Server** as Database Service
  + Optionally, use multiple storages, e.g., files, other Web services, databases (e.g. MySQL / MongoDB / Cassandra / etc.)
* Use **Entity Framework Core** to access your database
  + If you need additional connectors to other databases, feel free to use them
* Use **MVC Areas** to separate different parts of your application (e.g., area for administration)
* Adapt the default **ASP.NET Core site template** or get another free theme
  + Use responsive design based on **Twitter Bootstrap / Google Material design**
  + Or just design your own
* Use the standard **ASP.NET Identity System** for managing **Users** and **Roles**
  + Your registered users should have at least these two roles: **User** and **Administrator**
  + If you need, implement your own user management system
* Optionally, use **AJAX** request to asynchronously load and display datasomewhere in your application
* Write **Unit Tests** for your logic, controllers, actions, helpers, etc.
  + You should **cover** at least **65%** of your business logic implemented in the services.
* Implement **error handling** and **data validation** to avoid crashes when invalid data is entered
  + Both **client-side** and **server-side**, even at the database(s)
  + **Custom views** should be created for the following responses:
    - 404 Not Found Page
    - 500 Bad Request Page
* Handle correctly the special **HTML characters** and tags like **<br />** and **<script> (escape special characters)**
* **Use Dependency Injection**
  + **The built-in one in ASP.NET Core is perfectly fine**
* **Add pagination functionality. It is acceptable not having this functionality, only if the structure of the application doesn’t allow it.**
* **Add search or filtering functionality**
* **Optionally, use AutoМapping**
* **Seeding data:**
  + **Seed the database with relevant data.**
* **Prevent from security vulnerabilities like SQL Injection, XSS, CSRF, parameter tampering, AntiForgeryToken, etc.**
* **DO NOT use the project developed during the lectures by the lecturer. Try to do something different and implement your own idea.**

## Additional Requirements

Your Project **MUST** have a well-structured **Architecture** and a well-configured **Control Flow**.

* Follow the best practices for Object-Oriented design and **high-quality code** for the Web application:
  + Use the OOP principles properly: data encapsulation, inheritance, abstraction and polymorphism
  + Use exception handling properly
  + Follow the principles of strong cohesion and loose coupling
  + Correctly format and structure your code, name your identifiers and make the code readable
* Make the user interface (UI) good-looking and easy to use
  + If you provide a broken design, your Functionality Points will be sanctioned
* Support all major modern Web browsers
  + Optionally, make the site as responsive as possible – think about tablets and smartphones

## Source Control

Use a **source control system** by choice, e.g., **GitHub**, **BitBucket**

* Submit a link to your **public** source code repository
* You should have **committed** in at least **5 DIFFERENT** days
* You should have at least **20 commits**

**IMPORTANT:** The **Source Control Requirements** are **ABSOLUTELY MANDATORY**.   
**IMPORTANT: NOT** following the **Source Control Requirements** will result in your **DIRECT DISQUALIFICATION** from the **Project Defenses**.

## Submission Deadline

* You **must** submit your project before **23:59** on **13-August-2025** using a survey that will show up on **06-August-2025**.
* A presentation schedule will be available on **14-August-2025** and will include only the projects that were **submitted beforehand**. Non-submitted projects will **NOT** be evaluated.
* You can work on your project until **23:59** on **14th August**.

## Online Project Defense

Each student will have to deliver an online **defense** of his work in front of a trainer.   
Students will have **only 10-15 minutes** for the following:

* **Demonstrate** how the application works (very shortly)
* Show the **source code** and explain how it works
* Answer questions related to the project (and best practices in general)

Please be **strict in the timing**! On the 15th minute, you **will be interrupted**! It is a good idea to leave **the last 2-3 minutes for questions** from the trainers.

Be **well prepared** for presenting the maximum of your work for a minimum time. Open the project assets beforehand to save time.

## Bonuses

* Anything that is not described in the assignment is a bonus, if it has some practical use
* Examples
  + Use **SignalR** communication somewhere in your application
  + Use **Front-End Frameworks** (like **Angular**, **React**, **Blazor**)
  + Host the application in a **cloud environment**, e.g., in **AppHarbor** or **Azure**
  + Use a **file storage cloud API**, e.g., **Dropbox**, **Google Drive** or other for storing the files
  + Use of features of HTML5 like **Geolocation**, **Local Storage**, **SVG**, **Canvas**, etc.

## Assessment Criteria

* **Functionality** – **0…25**
* **Implementing controllers correctly** (controllers should do only their work) **– 0...5**
* **Implementing views and entities correctly – 0…10**
* **Unit tests** (unit testing **the services**, using mocking) **– 0…10**
* **Security** (prevent SQL injection, XSS, CSRF, parameter tampering, etc.) **– 0…5**
* **Data validation** (validation in the models and input models) **– 0…10**
* **Using areas with multiple layouts – 0…5**
* **Code quality** (services, well-structured code, following the MVC pattern, following SOLID principles, etc.) – **0…10**
* **Answer questions** related to the project (and best practices in general) – **0…20**
* **Bonus** (bonus points are given for exceptional project) – **0…20**