# ASP.NET Core – June 2021

# Individual Project Assignment 21-August-2021

## General Requirements

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

* The application must be implemented using **ASP.NET Core** Framework (**latest**).
  + Name – “**IT Asset Manager**”
  + User roles
    - Asset Administrator – Create Assets, Dispose Assets, Change Assets Status and Category, Delete Assets, CRUD Vendors, Addresses, Asset Categories, Asset Status, Departments
    - Asset User – Request Asset, Return Asset, View Assigned Assets, Search/View own Requests
    - User’s Manager – Approve/Reject User’s Requests, Search/View Requests for resp. Department
  + The application must have at least **10** web pages (views)
    - Register
    - Login
    - User pages:
      * Home – user’s assets list
      * Asset details
      * Request new asset
      * Request asset return
      * User’s requests
    - Manager pages:
    - Asset Administrator pages:
      * Home – Approved Requests for execution, Details -> Select Asset, Handover Asset
      * Requests - Search
      * Warehouses – Create, Edit, Delete
      * Create new asset
      * Assets list – Search, Details -> Edit, Dispose, Delete
      * Create new address – Country, Town, Address
  + The application must have at least 5 **entity models**
    - Users
    - Assets – (Id, Name, SerialNr, InventoryNr, CategoryId, Category, StatusId, Status, VendorId, Vendor, PurchaseDate, WarranyExpDate, DisposalDate, Invoice, Price, IsDeleted)
    - UsersAssets – (UserId, User, AssetId, Asset, AssignDate, ReturnDate, IsDeleted)
    - Categories - (Id, Name, ICollection<Asset> Assets, ICollection<Request> Request, IsDeleted)
      * Laptop, Desktop, Monitor, Printer, Projector
    - Status - (Id, Name, ICollection<Asset> Assets, IsDeleted)
      * In Use, In Stock, Pending Disposal, Disposed, Stolen, Missing
    - Requests – (Id, UserId, User, Details, SubmitDate, ApproveDate, HandoverDate, AssetCategoryId, Category, IsApproved, IsDeleted)
    - Vendors – (Id, Name, Vat, ICollection<Asset> Assets, IsDeleted)
    - Departments – (Id, Name, ManagerId, User Manager, ICollection<User> Users, IsDeleted)
  + The application must have at least 5 **controllers**
* Use **Visual Studio 2019 / JetBrains Project Rider**.
  + Use the **Razor** template engine for generating the UI
    - Use **sections** and **partial views**.
    - Use **display** and **editor templates**.
  + 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 one of these 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.
* 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)
* 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**
* **Optionally, use AutoМapping**
* **Prevent from security vulnerabilities like SQL Injection, XSS, CSRF, parameter tampering, etc.**
* **DO NOT use the project developed during the lectures by the lecturer. Try to do something different.**

## 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
* Use Caching where appropriate

## 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 **commits** in at least **5 DIFFERENT** days
* You should have at least **20 commits**
* You should **not commit** any changes after **23:59** on the **18-Aug-2021** (**Submission Deadline**)

**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 the **13-Aug-2021** using a survey that will show up on the **11-Aug-2021**.
* A presentation schedule will be available on the **19-Aug-2021** and will include only the projects that were **submitted beforehand**. Non-submitted projects will **NOT** be evaluated.

## Online Project Defense

Each student will have to deliver an online **defense** of its 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 timing**! On the 15th minute you **will be interrupted**! It is good idea to leave **the last 2-3 minutes for questions** from the trainers.

Be **well prepared** for presenting maximum of your work for 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…20**
* **Implementing controllers correctly** (controllers should do only their work) **– 0...5**
* **Implementing views correctly** (using display and editor templates) **– 0…5**
* **Unit tests** (unit test for some of the controllers 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 inversion of control – 0…5**
* **Using areas with multiple layouts – 0…10**
* **Code quality** (well-structured code, following the MVC pattern, following SOLID principles, etc.) – **0…10**
* **Bonus** (bonus points are given for exceptional project) – **0…25**