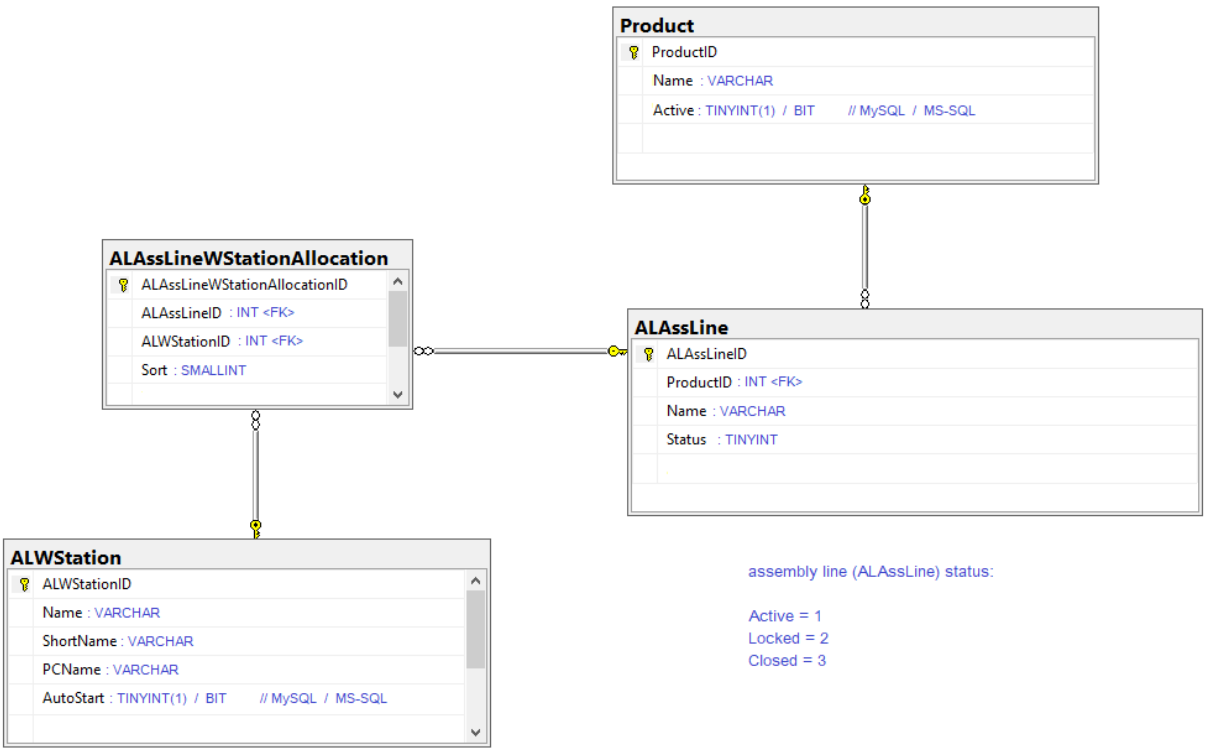
Create a simple application meeting the below criteria:

It should be a simple assembly lines manager for a switchgear assembly plant, allowing to input and manage products, assembly lines, workstations and allocate workstations to assembly lines. The relationships are like on the database diagram below: an assembly line belongs to a product, one product can have many assembly lines. One assembly line can have many workstations allocated to it, and one workstation can be allocated to many assembly lines.

Example entity names you can use in the app (so as not to waste time to make up your own names when testing your app):

- Products: *8DAB*, *8DJH*, *Simosec*, *NXPlus C*.
- Assembly lines: *Convey line*, *Manual line*, *Final assembly line*, *Testing line*
- Workstations: *Laser welding*, *Manual welding*, *Drive assembly*, *Voltage drop test*, *Leakage test*, *HV/PD test*, *Final inspection*, *Frame assembly*, *Dispatch*

The database should be implemented in MySQL or MS-SQL. The database schema should follow the below diagram:



Feel free to create additional tables if you need but the above is mandatory.

The app must be divided into separate projects: a backend (server side) and a frontend (client side). The frontend must be implemented in the Angular framework. The backend must be implemented in one of the following: Node.js, ASP.NET Core, ASP.NET MVC 5.

There should be a simple authentication implemented in the application. You can use an external user login API and integrate it into the application (a free API, e.g. Github Login API, so that I don't have to pay for anything to check your solution).

The use cases to implement:

1. Log-in a user
   Own solution or external API, it's your choice.
2. Display products with CRUD options.
   A simple list view showing all the products from the database. With CRUD buttons to manage the listed products.
3. Manage a product.
   A single product view allowing to set the product's properties like name, etc., and save the changes. After the save go back to the products list.
4. Display assembly lines with CRUD options.
   A simple list view showing all the assembly lines from the database with CRUD options. The list should contain a filtering by product.
5. Manage an assembly line.
   A single assembly line view allowing to set it's properties and save the changes. After the save go back to the assembly lines list. Remember that an assembly line must belong to a product.
6. Display workstations list with CRUD options.
   A simple list view showing all the workstations from the database with CRUD options.
7. Manage a workstation.
   A single workstation view allowing to set it's properties and save the changes. After the save go back to the workstations view.
8. Display assembly line – workstation allocations.
   A list showing assembly lines with allocated workstations, the layout can be as below:

Product	**Assembly line**	**Sort**	**Workstation short name**	**Workstation name**
8DJH	Convey line	1	LW	Laser welding
8DJH	Convey line	2	MW	Manual welding
8DJH	Convey line	3	DA	Drive assembly
8DJH	Convey line	4	LT	Leakage test
8DJH	Convey line	5	FI	Final inspection

   The list should contain filtering by product and by assembly line.
   The *Sort* column is the *ALAssLineWStationAllocation.Sort* which is the order of the workstation on the assembly line.
   It would be also nice to add an option to rearrange the order of workstations withing an assembly line (the *Sort*).
   The list view should contain two options (e.g., buttons):
   - Allocate workstation
   - Remove allocation
9. Allocate workstation.
   It should allow the user to allocate a selected workstation from the pool of workstations available in the database to a chosen assembly line. It's up to you how you realize this from the UI point of view. The outcome of this action should be a new record in the *ALAssLineWStationAllocation* table with the *Sort* value set to the next available ord-no., e.g., if the selected assembly line already had 6 worksations allocated then the new allocation's *Sort* is 7.
10. Remove allocation.
    On the allocations view a user selects one or more allocations on the list and clicks *Remove allocation*. Allocations are removed from the database and the view is refreshed.

Please share your solution with me anyway you like but the requirement is that it should be in the form of a git repository (or repositories) that I can easily clone. You can, e.g., put it on GitHub as a private repository and add me (*cezary-z*) to the collaborators.

It should be easy to set up and run. You can add a small readme to the repo if there are any caveats regarding running your solution.

Please also share with me a copy of the DB that you created, populated with some example data. Use any free file sharing service for example.

You don't have to complete it 100%. Do as much as you can, of course the more you complete the better but the goal is that I have a sample of your coding to estimate your experience and abilities.