Send us a solution (sourcecode) in an git repository in a ziparchive which contains the following (please do not include nuget packages or npm packages – we will do the dotnet restore and npm install ourself 😉) :

SolutionName:
Hahn.ApplicatonProcess.Application

Projects:
Hahn.ApplicatonProcess.July2021.Data - .Net 5.0 Class Library handles every DataAccess also HTTPDataAccess for assets, Uses RepositoryPattern and UnitOfWork Pattern
Hahn.ApplicatonProcess.July2021.Web- .Net 5.0 Kestrel Host, Hosts the static content as well as a WebAPI for the Frontend
Hahn.ApplicatonProcess.July2021.Domain – .Net 5.0 Class Library Containing Business Logic, Validators, Interfaces and Models
The solution must contain the needed docker files and docker-compose to run the application out of the box in docker environment.

We want you to build an Web Solution with RestEndpoints, as well as an Userinterface / Form to apply Data. Therefore you will have to separate the business logic, the data / persistence layer and the web project. The Api should have the following actions:
POST for Creating an Object – returning an 201 on successful creation of the object and the url were the object can be called
GET with id parameter – to ask for an object by id
PUT – to update the object with the given id
DELETE – to delete the object with the given id

The Objects we want to handle are the classes named User and Asset with the following properties:

Asset:
Id: string
    Symbol: string
    Name: string
User:
    Id: int
    Age: int
    FirstName: string
    LastName: string
    Address: string, or new table with the required fields as string
    Email: string
Assets: Asset[]

The WebApi accepts and returns application/json data.
The object and the properties should be validated by fluentValidation (nuget) with the following rules:
AssetName:
Id, symbol and Name must exist as valid combination on the API mentioned below:
– must be an existing asset (Show the user in the frontend only assets which are existing on the api endpoint: https://api.coincap.io/v2/assets but the ui shouldn't be slow) Doc: https://docs.coincap.io

User:

- Age: >18
- FirstName: At least 3 characters
- LastName: At least 3 characters
- Adress: Valid address with postal code, street, and house number (all fields are valid if the user entered values)
- Email: must be an valid email (only check for valid syntax *@*.[valid topleveldomain])

The main goal should be: A User can create his Profile and select some assets he wants to track. Always when the user opens his tracked asserts he should get the live data from the saved asserts of his profile.

If the object is invalid (on post and put) – return 400 and an information what property does not fullyfy the requirements and which requirement is not fullyfied.

Describe the API with swagger therefore use Swashbuckle v5 host the swaggerUI under [localhost]/swagger. Use https://github.com/mattfrear/Swashbuckle.Examples

Do not forget to provide example data in the SwaggerUI, so when someone click on try it out there is already useful valid data in the object that can be posted.

Use netcore logging to log each interaction with the API whereever it's meaningful to do so and also implement this
.AddFilter("Microsoft", LogLevel.Information)
.AddFilter("System", LogLevel.Error)

Write the log to a serilog rolling file sink the name needs to be setable in the applicationsettings.json file. Don't use serilog in Domain – if you want to log in domain project use https://www.nuget.org/packages/Microsoft.Extensions.Logging/

Frontend:
The including Form must be an Aurelia (http://aurelia.io) Application which uses the API to Post Data AND Validate all the inputs with the exact same parameters as the API does.
- use Typescript
- use Webpack
- Form can only be send if the data is valid
- Use Boostrap for the UI
- Use aurelia-validation, validate as much as you can :-)
- Use a Bootstrap FormRenderer
- invalid fields must be marked with an red border and an explanation why the date is invalid
- all strings must be using i18next
- the form has two buttons- send and reset.
- clicking the reset button an aurelia-dialog is shown - which ask if the user is really sure to reset all the data
- the reset button is only enabled if the user has typed in data -> if all fields are empty the reset button is not enabled.
- when the user has touched a field but afterwards deleted all entries, the reset button is also not enabled.
- The send button is only active if all required fields are filled out and are valid.
- after sending the data, the aurelia router redirects to a view which confirms the sending and shows the user all his created assets.

- if the sending was not successful an error message is shown in a aurelia-dialog. Describing what was going wrong.

For all strings you use, use localization and a Jsonfile as resource file.
To save the data use entityframework core 5.0 and entityframework in memory database.
If you have any questions – do not hesitate to ask. A Task like this is our daily business.
Provide an Readme.md how to build and start your application with the containing docker compose file.
Be creative and surprise us, additional features are getting extra points. But remember to match at least the requirements mentioned.