.NET/C# API case
In this assignment you can show what cool things you can build with .NET/C#. Shape this assignment the way you like best and make it as neat as you can.

Terms:
- This case revolves around building an API; you don't need to develop a front-end
- Use .NET 5.0 or higher
- Deliver the result as a GIT repository (preferably via Github, make sure it can be cloned by us)
- Provide a readme with instructions on how to start and use your project. Include the parts you are proud of and the parts you are less satisfied with and why you are proud/less satisfied.
- Deliver your endpoint documentation through a Swagger implementation in your project.

Part 1: API general
Generate a basic .NET Core Web API. The API should include the following features:
- Make sure the API uses a simple SQLite database (make sure the associated .db file is in your project)
- Create a database model for address information (street, house number, zip code, city, country)

- Create a controller with the following endpoints:
- GET /addresses For retrieving multiple addresses
- GET /addresses/:id To retrieve a single address
- POST /addresses To add a new address to the database.
- PUT /addresses/:id To edit an address in the database.
- DELETE /addresses/:id To remove an address from the database.

- Make sure when creating and editing your address that all fields are mandatory and will be validated on this

Part 2: Filters
Add functionality to the GET /addresses endpoint so you can search and sort the result and make sure these options can be applied by query parameters.
- Search should be possible by specifying a search value that searches all address fields for matches
- Sort should be possible on all address fields in ascending and descending order

TIP: Try to implement this as neatly (and generically) as possible. Try to avoid using a large LINQ query / switch / if statement for filtering and sorting. Find a way to dynamically search fields from the model, so that if an address field were ever added, you don't have to add extra code for filtering and sorting. Of course you can use a package for this, but ultimately we'd rather see how you would solve this with your own code :)

Part 3: Distances
Connect to a public geolocation API, which allows to calculate distances between two addresses from the previously created database. To do this, create an additional endpoint

that can retrieve the distance from an address A to another address B (in kilometers). Give this your own interpretation to implement this as neatly as possible.

You can't manage, or you don't have enough time to implement all parts? No problem. Then indicate in your readme how you would do it, and explain your thinking well.