.NET Test Task Description

1. Use C#, ASP.NET Core (any version), EntityFramework Core (Code First)

2. Create ASP.NET Core API that will be available via Swagger:
   https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio
   Create from scratch or use any code boilerplate/template from Microsoft or like:
   https://aspnetboilerplate.com/
   NOTE: do not need to create any UI by yourself

3. Create Database (any MSSQL or MySql or else) for storing entities:

```
public class Person
{
        public long Id { get; set; }

        public string FirstName { get; set; }
        public string LastName { get; set; }

        public long? AddressId { get; set; }
        public virtual Address Address { get; set; }
}

public class Address
{
        public long Id { get; set; }
        public string City { get; set; }
        public string AddressLine { get; set; }
}
```

4. Create a Person API service that with 2 endpoints/methods
   4.1 API endpoint to store person from string into database

```
public Task<long> Save(string json)
{
        // deserialize string into object of type Person using own deserializer (see more details in
Minimal requirements section)
        // DO NOT use 3rd party libraries for deserialization like Json.NET or Microsoft
        // insert or update Person entity in database
        // return entity id
}
```

   4.2 API endpoint to query persons from database

```
public class GetAllRequest
{
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string City { get; set; }
}


public Task<string> GetAll(GetAllRequest request)
{
```

```
        // get Persons entities from database
        // filter by GetAllRequest fields (null or empty fields should be ignored)
        // use your own manually written json serializer to serialize result into string
}
```

Implement next use case

1. Call Save api endpoint with next json string:
   ```
   {
        firstName: 'Ivan',
        lastName: 'Petrov',
        address: {
            city: 'Kiev',
            addressLine: prospect "Peremogy" 28/7,
        }
   }
   ```
   // NOTE that there is no " (quote) char in 'addressLine' field value before 'prospect' and after '7', so your json deserializer should handle such cases

2. Call GetAll api endpoint with GetAllRequest json string as request parameter:
   ```
   {
        "city": "kiev"
   }
   ```
   So request parameters will be used as filtering all entities in the database.
   If multiple filters are set, for example city and first name, then we should filter by city AND first name.
   Note that api results should contain a unique id for each item. Then when you call the Save method this id will be used to determine whether we need to create or update an entity.

3. Use the result of previous GetAll api for calling Save api endpoint.

4. Call GetAll api endpoint with GetAllRequest empty object json string:
   ```
   {
   }
   ```

5. The result of step 2 and 4 should be equal. Steps from 2 to 4 are needed to ensure that you implemented the save method correctly and you do not duplicate data in the database and properly handle saving entities with id.

Minimal requirements:

- Use Repository pattern.
- Implement your own/custom **from scratch** simple Json serializer / deserializer.
  NOTE: do not need to support/implement types that are not present in test classes (see above) like Arrays, Dictionary, List etc. Do not use 3rd party libraries/code: System.Text.Json.Serialization, Json.NET and others. Should be reusable (able serialize any type), so input parameter is object.
- Use **Dependency Injection.** Use one of the IOC containers: AutoFac, Ninject, Castle Windsor, Unity, Microsoft (built-in in ASP.NET Core) etc.

The result of the completed task can be provided as a link to the github project or a link to a file for downloading or any other convenient method.
You do not deploy the result of a test task to any hosting like AWS, Azure, else.

Deadline: **up to 7 days** (**can be extended**).