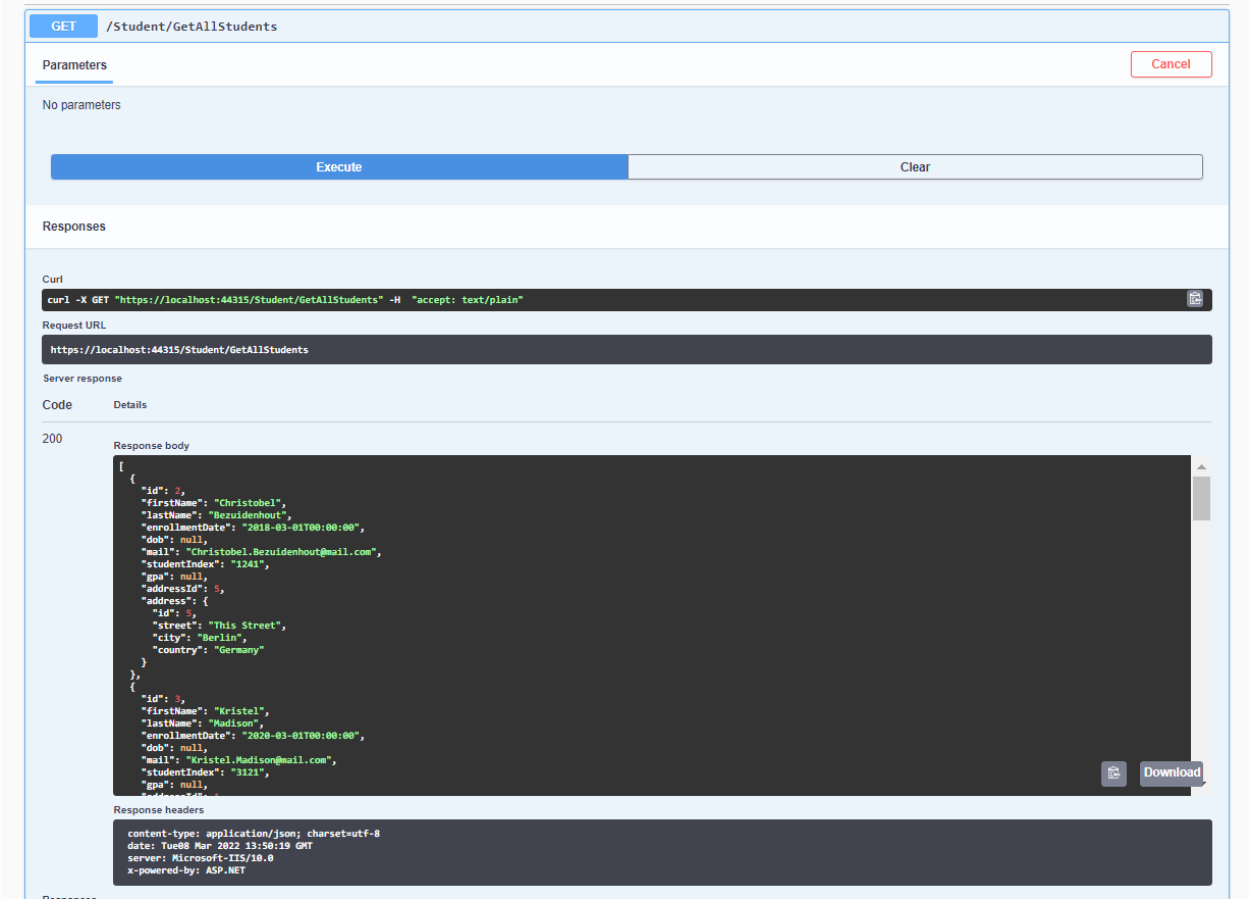
Internet Services Part-Time Exam

Instructions: Create a simple Staff Management Application. Before you start, read the instructions to the end.

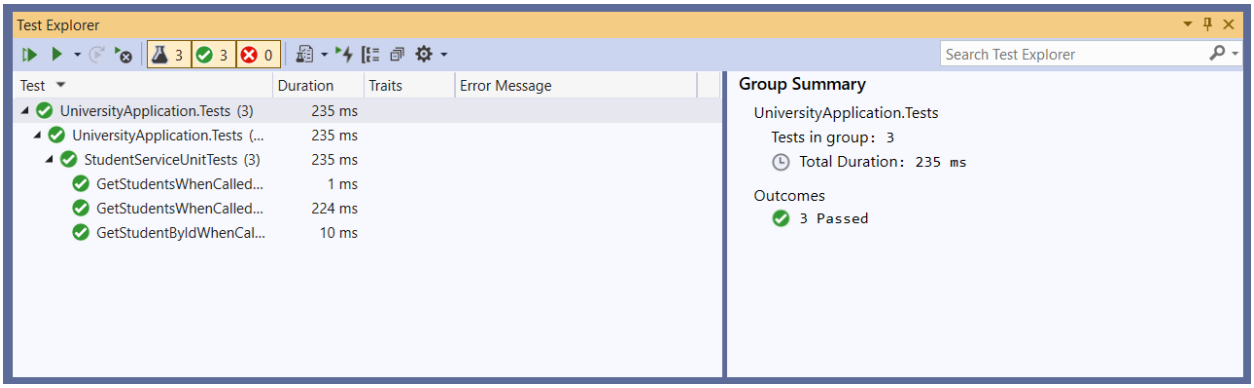
Tasks:

- Create a Web Api solution using .Net 8.
- All data in your solution should be persisted in a database. Use Entity Framework Core Code first approach.
- Create a layered architecture following the Service-Repository pattern and implement the requirements in the 3 layers appropriately: Data, Service and WebApi layers (feel free to use other naming convention for your layers as preferable).
- Create the necessary entities: Employee and Company.
- Employee should have the following properties: Autogenerated PK Id, First Name, Last Name, DOB, Address, Employment Start Date, Total Compensation, Notice Period and FK Company Id which will be the connection to the Companies table.
- On the Employee entity set the following restrictions: Employee Id should be autogenerated, the properties: First Name, Last Name, DOB, Address, Employment Start Date, Total Compensation and Notice Period should be required. First Name should have max length of 100 chars and Last Name should have max length of 200 chars, Address should have max length of 300 chars.
- Company should have the following properties: Autogenerated PK Id, Name, Address, Owner, Year, Number of Employees.
- On the Company entity set the following restrictions: Company Id should be autogenerated, all other properties: Name, Address, Owner, Year, Number of Employees should be required.
- Each Employee is allowed to work in one Company at a time only, while the same Company can have multiple Employees (one Company can relate to multiple Employees).
- Create the two appropriate interfaces and define methods get all, get by Id, create, update and delete for Employee and Company.
- Create DTOs for the Employee and Company.
- Create two services EmployeeService and CompanyService and implement the interfaces IEmployeeService and ICompanyService for Employee and Company.
- Create two API controllers EmployeeController and CompanyController with CRUD methods using the services and add appropriate implementation using DI and IOC.
- Implement the EmployeeRepository and CompanyRepository.
- Implement the two interfaces IEmployeeRepository and ICompanyRespository respectively in the correct layer.

- Use the created repositories and appropriate interfaces to communicate and manipulate the data in the database using DI and IOC.
- Create a Test project and implement a few Unit tests.
- For the EmployeeService create at least 3 Unit tests.
- For the CompanyService create at least 3 Unit tests.
- Run the Unit tests and make sure that they run successfully.
- Put the source code in a Zip folder named Name_Surname_Id.zip and upload on the provided submission link. If the solution is too big for the submission link, you can upload it on github or cloud and upload a .txt file containing a link to your online repository on the provided submission link. **Make sure that your repository is publicly accessible!**
- Together with the source code, provide screenshots of all the endpoints for both Employee and Company: Get All Employees, Get Employee, Post Employee, Put Employee, Delete Employee and Get All Companies, Get Company, Post Company, Put Company, Delete Company. Screenshots could be from either Swagger or Postman. Screenshots should contain both visible **request** and **response**. **Screenshots are mandatory! No screenshots = no points!** Example of how one screenshot should look can be found below:



- Together with the source code, provide screenshots of the Test Explorer Run after you have ran them all. **Screenshots are mandatory! No screenshots = no points!** Example of how one screenshot from the test runs should look can be found below:



Requirements & Tips:

- You should use .Net 8 version.
- You can use AutoMapper for models mapping. Using AutoMapper is advisable, but you can also implement manual mapping for the entities and DTOs.
- For data layer you should use Entity Framework Core.
- You can add some test data to the database, but it is not mandatory.
- You can use either Swagger or Postman for testing your web api and for providing the test screenshots.
- Providing the backup from your database is not mandatory.