Department of Information Systems and Technologies

Name:	
Surname:	
ID Number:	
Section:	

CTIS 465

Microservice Development with .NET

SPRING 2024 – 2025 FINAL 180 minutes Total points: 100 17.05.2025

Solutions which contain compile errors will not be graded!

1. **(1 points)** Clone the **ETradeMS** repository from the GitHub link below: https://github.com/cagilalsac/ETradeMS

2. **(5 points)** Modify the **ProductsDbHandler** class in **APP.Products** project's **Features** folder which inherits from the **Handler** class of the **CORE** project and provides the injection of the DbContext of type **ProductsDb** through the constructor, also sending the United States culture (en-US) as parameter to the base **Handler** class.

3. **(4 points)** Modify the **UsersDbHandler** class in **APP.Users** project's **Features** folder which inherits from the **Handler** class of the **CORE** project and provides the injection of the DbContext of type **UsersDb** through the constructor, also sending the United States culture (en-US) as parameter to the base **Handler** class.

4. **(55 points: 5 points for each class)** Create the request, response and handler classes in **APP.Products** project's **Features** folder as below:

   - **Categories Folder**
     - **CategoryQueryResponse : QueryResponse**
       - **Name string**

   - **Stores Folder**
     - **StoreQueryResponse : QueryResponse**
       - **Name string**

   - **Products Folder**
     - **ProductQueryRequest : Request, IRequest<IQueryable<ProductQueryResponse>>**

- **ProductQueryResponse : QueryResponse**
  - **Name string**
  - **UnitPrice decimal**
  - **ExpirationDate DateTime?**
  - **IsDiscontinued bool**
  - **CategoryId int**
  - **StoreIds List<int>**
  - **UnitPriceF string (UnitPrice value formatted as “C2”)**
  - **ExpirationDateF string (ExpirationDate value formatted as “MM/dd/yyyy”)**
  - **IsDiscontinuedF string (IsDiscontinued value formatted as “Yes” or “No”)**
  - **CategoryName string**
  - **Category CategoryQueryResponse**
  - **StoreNames string**
  - **Stores List<StoreQueryResponse>**

- **ProductQueryHandler : ProductsDbHandler, IRequestHandler<ProductQueryRequest, IQueryable<ProductQueryResponse>>**

  - **public Task<IQueryable<ProductQueryResponse>> Handle(ProductQueryRequest request, CancellationToken cancellationToken);**

  This method first orders products descending by **IsDiscontinued** then ascending by **Name**, projects **Product** entity properties to **ProductQueryResponse** properties and returns the **ProductQueryResponse** query.

- **ProductCreateRequest : Request, IRequest<CommandResponse>**
  - **Name string, required with maximum 150 characters**
  - **UnitPrice decimal**
  - **ExpirationDate DateTime?**
  - **IsDiscontinued bool**
  - **CategoryId int**
  - **StoreIds List<int>**

- **ProductCreateHandler : ProductsDbHandler, IRequestHandler<ProductCreateRequest, CommandResponse>**

  - **public async Task<CommandResponse> Handle(ProductCreateRequest request, CancellationToken cancellationToken);**

  This method first checks if there are any products with the same name exists in the **Products** table and if exists returns an **error CommandResponse** with message *“Product with the same name exists!”*. If a product with the same name doesn’t exist, creates a new product in the **Products** table and returns a **success CommandResponse** with message *“Product created successfully.”*.

- **ProductUpdateRequest : Request, IRequest<CommandResponse>**
  - **Name string, required with maximum 150 characters**
  - **UnitPrice decimal**
  - **ExpirationDate DateTime?**
  - **IsDiscontinued bool**
  - **CategoryId int**
  - **StoreIds List<int>**

- **ProductUpdateHandler : ProductsDbHandler, IRequestHandler<ProductUpdateRequest, CommandResponse>**

  - **public async Task<CommandResponse> Handle(ProductUpdateRequest request, CancellationToken cancellationToken);**

  This method first checks if there are any products other than the product of the request with the same name exists in the **Products** table and if exists returns an **error CommandResponse** with message *"Product with the same name exists!"*. If a product with the same name doesn't exist, gets the product entity from the **Products** table. If the product entity is not found, returns an **error CommandResponse** with message *"Product not found!"*. If the product entity is found, first deletes the relational **ProductStores** data then updates the entity properties from the request properties and commits changes to the database. Finally, returns a **success CommandResponse** with message *"Product updated successfully."*.

- **ProductDeleteRequest : Request, IRequest<CommandResponse>**

- **ProductDeleteHandler : ProductsDbHandler, IRequestHandler<ProductDeleteRequest, CommandResponse>**

  - **public async Task<CommandResponse> Handle(ProductDeleteRequest request, CancellationToken cancellationToken);**

  This method first gets the product entity from the **Products** table. If the product entity is not found, returns an **error CommandResponse** with message "*Product not found!"*. If the product entity is found, first deletes the relational **ProductStores** data then deletes the product entity and commits changes to the database. Finally, returns a **success CommandResponse** with message *"Product deleted successfully."*.

5. **(8 points: 4 points for each method)** In **UsersDbHandler** class within the **Features** folder of the **APP.Users** project:

    a. Implement
    protected virtual string CreateAccessToken(List<Claim> claims, DateTime expiration);
    method that returns JWT.

    b. Implement
    protected virtual List<Claim> GetClaims(User user);
    method that returns claims from a user entity.

6. **(12 points: 4 points for each class)** Create the request, response and handler classes In **APP.Users** project's **Features** folder as below:

- **Users Folder**
    - **TokenRequest : Request, IRequest<TokenResponse>**
        - **UserName string, required with maximum 30 minimum 3 characters**
        - **Password string, required with maximum 15 minimum 3 characters**

    - **TokenResponse : CommandResponse**
        - **Token string**

    - **TokenHandler : UsersDbHandler, IRequestHandler<TokenRequest, TokenResponse>**

        - **public async Task<TokenResponse> Handle(TokenRequest request, CancellationToken cancellationToken);**

        This method first gets the user entity from the **Users** table from request's **UserName** and **Password** properties for only user entity's **IsActive** property value **true**. If the user entity is not found, returns an **error CommandResponse** with message *"Active user with the user name and password not found!"*. If user entity is found, gets the user entity claims and assigns them to a **claims** variable by invoking the **GetClaims()** method of the base class. Then assigns an **expiration** variable by adding minutes defined in **ExpirationInMinutes** property of the **AppSettings** class to the current date and time. Finally, returns a **successful TokenResponse** instance with *"Token created successfully."* message assigning the property value **Token** by invoking the **CreateAccessToken()** method of the base class.

7. **(10 points)** In **API.Users** project:

    a. Create an API controller named **UsersController** and implement a **post Token** action which returns a JWT within the **TokenResponse** with **OK Http Status Code** if sending the **TokenRequest** to the mediator and getting the **TokenResponse** is successful. If not successful, return **Bad Request Http Status Code** with **TokenResponse** instance's message.

    b. The route of the **Token** action must be *"api/Token"*.

    c. Anyone without any authentication can send a request to the **Token** action.

    User with role Admin information**:**
    User Name: **admin**
    Password: **admin**

    User with role User information**:**
    User Name: **user**
    Password: **user**

8. **(5 points)** In **API.Projects** project:

    a. Create an API controller named **ProductsController** and implement **Get**, **Get by Id**, **Post**, **Put** and **Delete** actions.

    b. Any authenticated user can send a request to **Get** and **Get by Id** actions. However, only users with role **Admin** can send a request to **Post**, **Put** and **Delete** actions.