PE\_PRN231\_FA23\_TrialTest  
FALL 2023  
Subject: PRN231  
Duration: 90 minutes

**INSTRUCTIONS**

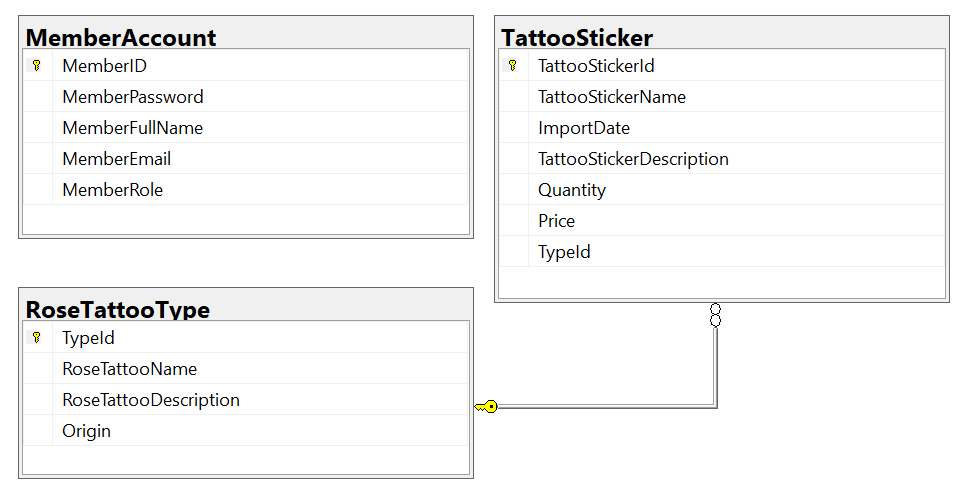
**Please read the instructions carefully before doing the questions.**

* You are **NOT allowed** to use any device to share data with others.
* You must use **Visual Studio 2019 or above, MSSQL Server 2012 or above** for your development tools.

**IMPORTANT – before you start doing your solution, MUST do the following steps:**

1. Create Solution/Project in Visual Studio named **PE\_PRN231\_FA23\_TrialTest\_StudentCode**.
2. *Client Project named:* ***PE\_PRN231\_FA23\_TrialTest\_StudentCode\_Client.sln*** (set the default Client application for your project as **Login** page) *or* ***PE\_PRN231\_FA23\_TrialTest\_StudentCode\_PostmanScript.json***
3. Create your MS SQL database named **RoseTattoo2023DB** by running code in script **RoseTattoo2023DB.sql.**
4. *You are not allowed to connect direct to database from ASP.NET Core Web API, every database connection must be used with Repository and Data Access Objects.*
5. *The database connection string must get from appsettings.json file.*
6. *Note: Please check before you submit your work, no syntax error(s) in the submitted PE.* ***If there are syntax errors or compilation errors in your PE program, you will not pass the PE requirements, the mark will be 0.***

Implement a part of Tattoo Sticker Store Application. The below Figure is a part of **RoseTattoo2023DB** database.



Note that: *Member role: Administrator = 1; Staff = 4; Member = 2, Manager=3.* A rose tattoo type will have many tattoo stickers, one tattoo sticker will belong to only one rose tattoo type.

**Part 01 (7.0 marks)**. Create an API using **ASP.NET Core Web API** with Entity Framework Core. A MS SQL Server database will be created to persist the data and it will be used for reading and managing data.

1. Check authentication/authorization with the ASP.NET Core Web API with JSON Web Token (JWT), ***only manager role will have a permission to to all actions (CRUD and search). Note that the permission for Login process is using MemberEmail and MemberPassword.***
2. You must use OData to implement the ASP.NET Core Web API. CORS is using in this case.

* Implement CRUD actions with **TattooSticker** table. Add new item with the r*equirements:*
  + All fields are required.
  + Value for TattooStickerName includes a-z, A-Z, /, \*, $, #, space and digit 0-9. Each word of the TattooStickerName must begin with the capital letter.
  + Value for ImportDate >=1990 and ImportDate <= current date.
* Get all tattoo sticker information with specific TattooStickerId(the information includes TattooStickerName, TattooStickerDescription, ImportDate, Quantity, Price and *RoseTattooName*).
* Search all tattoo stickers with 2 conditions: TattooStickerDescription or ImportDate (using relative search)

**Part 02 (3.0 marks).**

Student can choose one of these options for client application

* *GUI client app*
* *Postman test cases*

***Option 1. GUI Client application***

Student can choose one of these options for client application

* *ASP.NET Core Web App with MVC*
* *ASP.NET Core Razor Pages*
* *Windows Forms*
* *Windows Presentation Foundation*

1. Authentication function

If user with a *manager role* logs in successfully, save this information to a temporary parameter. All CRUD actions are required with authentication. In the case login unsuccessfully, display *“You are not allowed to access this function!”*

1. Check if login successfully, search all *tattoo stickers* of specific TattooStickerId with 2 conditions: ImportDate or TattooStickerDescription (OR option for this case).
2. Check if login successfully, delete the selected item with the confirmation then update the list of tattoo stickers.
3. Check if login successfully, add new item with the r*equirements:*

* All fields are required.
* Value for TattooStickerName includes a-z, A-Z, /, \*, $, #, space and digit 0-9. Each word of the TattooStickerName must begin with the capital letter.
* Value for ImportDate >=1990 and ImportDate <= current date.

***Option 2. Postman Script***

Using Postman application to write and test auto endpoint in Web API.

* Each test case have 3 basic steps:
  + Send Http request to API endpoint
    - Add JWT token in header
    - Fill parameter for request
    - Use correct HTTP verb
  + Get Http response and check
    - Return status code
    - Message body return
  + Return pass/fail result of test case
* You have to write at least 6 test cases
  + Login successful/Login failed
  + Create **TattooSticker**
  + Update **TattooSticker**
  + Delete **TattooSticker**
  + Get **TattooSticker** by Id
  + Get list of TattooStickerDescription or ImportDate