ASHBOURNE

Gym Management Exercise – Instructions

Context:

The aim of this exercise is to allow you to demonstrate your style and approach to key concepts by building a simple in-memory Gym Membership System using C# and WinForms.

The aim of the system is to help gym staff manage members and their membership types, simulate sign-ups, and generate basic reports.

Deadline:

Please submit your work by latest 17:00 Tuesday 11th November.
This should be by E-mail to development@ashbournemanagement.co.uk.

Setup:

You will be provided with a pre-configured **Visual Studio 2022 WinForms project**, targeting **.NET 7.0**. If this framework is not currently installed on your machine, you'll need to download and install it from the official .NET downloads page.

The solution includes basic scaffolding to help you focus on the core tasks rather than spending time on boilerplate setup. All necessary files are included to get you up and running quickly — including UI stubs, models, and mock data storage.

If you feel that adding a **NuGet package** (e.g., for validation, better UI components, or utility functions) would improve your implementation, you are welcome to do so. Just be sure to:

- Clearly list any added packages in your `README.md`
- Briefly explain why you included each one

Objectives:

The exercise is split into 4 main areas:

1. **WinForms UI Implementation**
2. **Model and Data Structure Design**
3. **Logic and Validation**
4. **LINQ and SQL Simulation**

Tasks

1. ***WinForms UI Implementation***

- Create a `MainForm` with navigation (e.g., using tabs or buttons) to access the Member Management, Membership Management, and Report views.
- Implement a `MemberForm` to display a list of all members using a `DataGridView`. Add functionality to: ○ Add a new member via a form input. ○ Edit existing member details. ○ Delete a selected member.
- Implement a `MembershipTypesForm` to manage membership types:
  ○ Display all membership types with `DataGridView`. ○ Add/Edit/Delete membership types with relevant fields (e.g., Name, Monthly Fee, Initial Fee, Max Members).
- Provide UI controls for assigning a member to a membership type, ensuring a member can only be signed up once.
- Implement a `ReportForm` to show a visual summary of sign-ups, groupings, and totals using a `RichTextBox` or `DataGridView`.

2. ***Model and Data Structure Design***

- A `Member` class has been defined for you with the following properties: `MemberId, FirstName, LastName, DateOfBirth, Email, and MembershipType.`
- Define a `MembershipType` class with properties: `MembershipTypeId, Name, MonthlyFee, InitialFee and MaxMembers`.
- Define a `SignUp` class with properties: `Member` and `MembershipType`.
- Store all data in-memory using `List<T>` collections in a static `GymDataStore` class.

3. ***Logic and Validation***

- Prevent the addition of members with duplicate email addresses or identical name + email combinations.
- Validate all member input fields:
  - First/Last name required. ○ Date of Birth must be valid between ages 16-100.
  - Email must follow a proper format.
- Enforce the rule that the number of Signups in a `MembershipType` cannot exceed its `MaxMembers`.

- Prevent assigning a member to multiple membership types or duplicate signups.

*4. **LINQ and SQL Simulation***

Simulate the following SQL-style queries using LINQ on your in-memory collections:

- Display a list of all members where their Age `> 30`, ordered by last name.
- Show all membership types with the total number of sign-up members (simulate `GROUP BY`).
- Join members and membership types to create a summary list with member name, email, and their assigned membership type.
- Display a count of how many members are assigned to each membership type.

5. Mock API Simulation

To simulate basic front-end/backend interaction and JSON handling, your solution must include:

- A MockApiService class that:
  - Contains embedded JSON strings representing sample Member and MembershipType data.
  - Mimics asynchronous API calls using Task.Delay and deserialization via System.Text.Json.
- A simple UI (e.g., in MainForm) that:
  - Loads data from the mock API service.
  - Displays a summary report using a DataGridView.
  - Joins members and membership types to show name, email, and membership type.

Bonus (Optional)

- Add functionality to export the report to a `.txt` or `.csv` file using `StreamWriter`.
- Implement an "Undo Delete" feature that temporarily stores the last deleted member and allows it to be restored.
- Use `BindingSource` to handle clean binding between your data lists and the WinForms UI.
- Add color coding or minimal styling to improve user readability in the UI.

Submission

- Submit a ZIP of your completed Visual Studio solution.
- Include a `README.md` with: ○ A short description of your solution. ○ How to run it. ○ Any assumptions you made.
  ○ Screenshots (optional but encouraged).

Evaluation Criteria

Category	
Code Quality	30%
Correctness/Features	30%
UI Implementation	20%
LINQ/SQL Simulation	20%