# Develop a WEB API Project for Gym Management

**Problem Statement:**

Develop a WEB API project for a gym management system using ASP.NET Core. The API will provide services for managing gym classes and members. Your task is to design and implement the API based on the given requirements, focusing on action methods, controllers, endpoints, and appropriate status codes.

**Models:**

**Class.cs:**

* **ClassId**: An integer representing the unique identifier for each class (auto incremented).
* **ClassName**: A string representing the name of the class.
* **Description**: A string containing a brief description of the class.
* **Schedule**: A string representing the schedule for the class.
* **Instructor**: A string representing the instructor for the class.
* **Capacity**: An integer representing the maximum number of members allowed in the class.
* **Members (ICollection<Member>)**:
* A collection of Member entities associated with the class. This represents a one-to-many relationship where one class can have multiple members enrolled. It is optional.
* This property is marked with [JsonIgnore] to prevent circular references during JSON serialization.

**Member.cs:**

* **MemberId**: An integer representing the unique identifier for each member (auto incremented).
* **FullName**: A string representing the full name of the member.
* **Email**: A string representing the email address of the member. It must be in a valid email format.
* **Phone**: A string representing the phone number of the member.
* **MembershipType**: A string indicating the type of membership (e.g., regular, premium).
* **ClassId**: An integer representing the foreign key linking to the Class entity. This establishes a many-to-one relationship where multiple members can be associated with one class.
* **Class**:
* A reference to the Class entity associated with the member. It is optional.
* This property is marked with [JsonIgnore] to prevent circular references during JSON serialization.

Using **ApplicationDbContext** for **Class** and **Member** Management. **ApplicationDbContext** must be present inside the **Data** folder.

**Namespace - dotnetapp.Data**

The ApplicationDbContext class acts as the primary interface between the application and the database, managing Create & Delete operations for **Class** entities and Create & Read operations for **Member** entities. This context class defines the database schema through its DbSet properties and manages the **one-to-many relationship** between Class and Member entities using the Fluent API.

**DbSet Properties:**

* **DbSet<Class> Classes**:
* Represents a collection of Class entities stored in the Classes table. Each class can have multiple associated Member entries, defining the one-to-many relationship between Class and Member (i.e., one class can have many members).
* **DbSet<Member> Members**:
* Represents the Members table in the database. Each member can be associated with a single class, establishing the many-to-one relationship between Member and Class using the ClassId foreign key.
* **OnDelete(DeleteBehavior.Cascade)**: This is where the deletion behavior is configured. **Cascade delete** means that when a Class entity is deleted, all associated Member entities will also be automatically deleted from the database.

**Implementing the Logic in the Controller:**

**Controllers: Namespace: dotnetapp.Controllers**

**ClassController**:

* **CreateClass([FromBody] Class gymClass)**:
* Creates a new class to the database.
* Make a POST request to **/api/Class** with the data in the request body.
* Upon successful creation, it returns a 201 Created with the location of the newly created class.
* Return Type: **Task<ActionResult>**
* **DeleteClass(int id)**:
* Deletes the class identified by id.
* Make a DELETE request to **/api/Class/{id}** where {id} is the ID of the class.
* If the class is not found, it returns a 404 Not Found.
* Upon successful deletion, it returns 204 No Content.
* Return Type: **Task<ActionResult>**

**MemberController**:

* **CreateMember([FromBody] Member member)**:
* Creates a new member to the database.
* Make a POST request to **/api/Member** with the data in the request body.
* Upon successful creation, it returns a 201 Created with the location of the newly created member.
* Return Type: **Task<ActionResult>**
* **GetAllMembers()**:
* Retrieves a list of all members along with their associated classids.
* Make a GET request to **/api/Member**
* If no members are found, it returns a 204 No Content.
* Otherwise, it returns a 200 OK with the list of members.
* Return Type: **Task<ActionResult>**
* **GetMemberById(int id)**:
* Retrieves a single member by their MemberId.
* Make a GET request to **/api/Member/{id}** where {id} is the ID of the member.
* If the member is not found, it returns a 404 Not Found.
* If found, it returns a 200 OK with the member details.
* Return Type: **Task<ActionResult>**

**Endpoints:**

**Classes**:

* **POST /api/Class**: Create a new class.
* **DELETE /api/Class/{id}**: Delete a class by its ID.

**Members**:

* **POST /api/Member**: Create a new member.
* **GET /api/Member**: Retrieve a list of all members, including their associated classes.
* **GET /api/Member/{id}**: Retrieve a specific member by their ID.

**Status Codes and Error Handling:**

* **204 No Content**: Returned when no records are found for classes or members.
* **200 OK**: Returned when records are successfully retrieved.
* **201 Created**: Returned when a new class or member is successfully created.
* **400 Bad Request**: Returned when there are validation errors or missing required fields during creation.
* **404 Not Found**: Returned when a class or member is not found during retrieval or deletion.

**Note:**

* Use swagger/index to view the API output screen in 8080 port.
* Don't delete any files in the project environment.