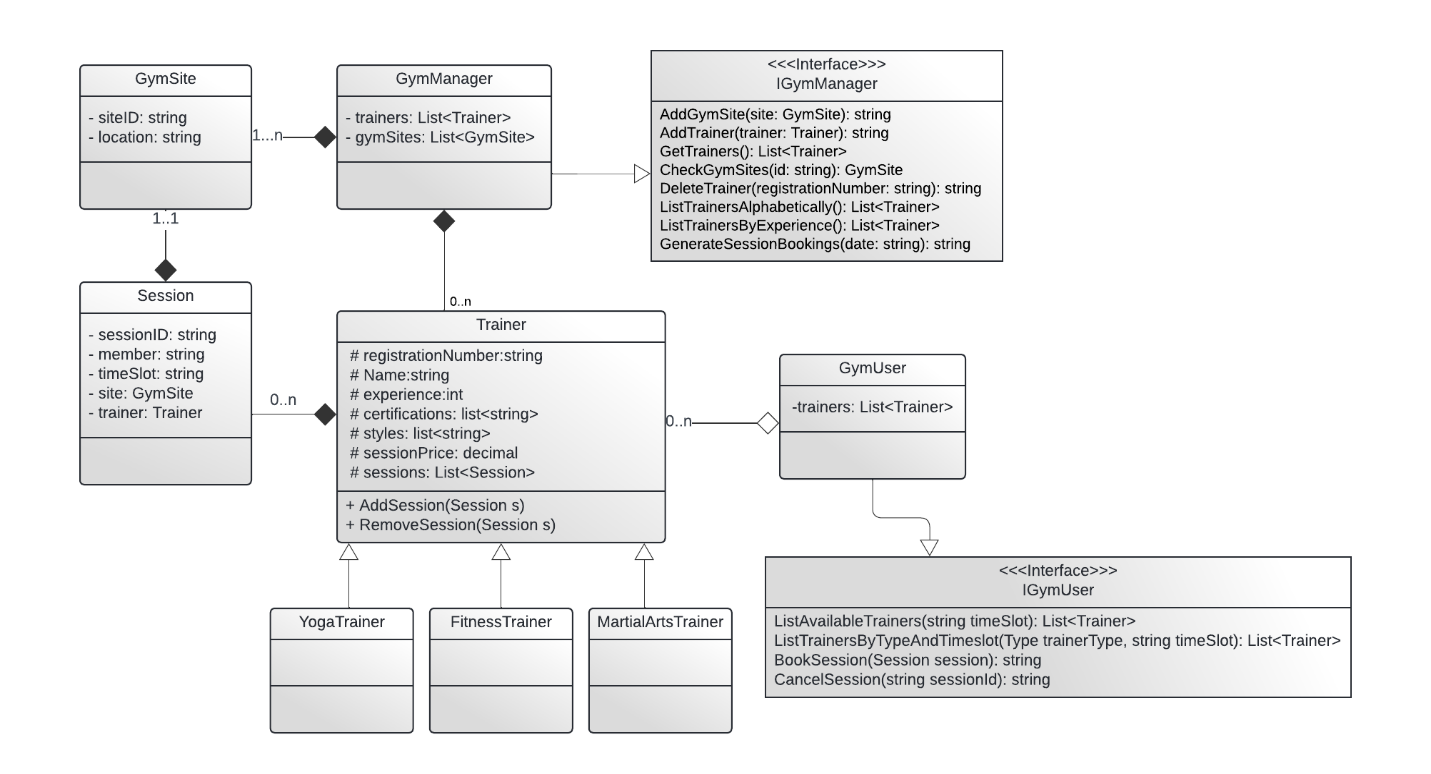
**Project Design Report: Gym Management System**

# Introduction:

The Gym Management System is a console application in C# .NET to manage multiple gyms across London efficiently. It focuses on managing personal trainers and booking and cancellation of training sessions for members. The system consists of two main user roles: Admin and Gym Member (User), each with specific functions. Admins can add gym sites, manage trainers, view them sorted by name and experience and generate reports and Gym Members can book sessions and view available trainers.

# UML Diagram:

Here is an overview of UML diagram of the gym management system:



## Relations:

* GymManager implements IGymManager Interface.
* GymManager has composition relation with Trainer containing a list from 0 to many.
* GymManager has composition relation with GymSite containing a list from 1 to many.
* Trainer has composition relation with Session containing a list from 0 to many.
* Inheritance between Trainer and subclasses YogaTrainer, MartialArtsTrainer and Fitnesstrainer.
* Session has an association having one gymsite at a time and cannot exist without it.
* GymUser implements IGymUser Interface.
* Trainer has an association with GymUser having a list of it.

# Class Descriptions:

Here is an overview of classes and interfaces implemented in the application:

## GymManager Class:

GymManager class implements the interface IGymManager and is responsible for the functionalities of the admin of the application and allows him to perform various tasks.

### Attributes:

It consists of two lists. One of trainers and second one consists of gymsites.

### Methods:

It consists of multiple methods such as AddGymSite, CheckGymSite, AddTrainer, DeleteTrainer, ListTrainersAlphabetically, ListTrainersByExperience, GenerateSessionBookings.

## GymUser Class:

GymUser class implements the interface IGymUser and is responsible for the functionalities of the member user of the application and allows him to perform various tasks.

### Attributes:

It consists of One of trainers.

### Methods:

It consists of multiple methods such as ListAvailableTrainers, ListTrainersByTypeAndTimeSlot, BookSession and Cancel Session.

## Trainer Class:

It is a base class with come common attributes for derived.

### Attributes:

It consists of string: registration number which is unique identifier, name, certifications, styles, experience, session price and a list of sessions of trainer.

### Methods:

It consists of multiple methods such as GetRegistrationNumber() and GetName() etc.

## FitnessTrainer, YogaTrainer and MartialArtsTrainer Classes:

These are the derived classes of trainer class.

## Session Class:

Session class is a class that stores users information related to the session booked.

### Attributes:

It consists of string: sessionID which is unique identifier, member, trainer, timeslot and the gymsite allocated for the training.

## GymSite Class:

It stores info about the different gym sites across area for a gym.

### Attributes:

It consists of string: siteID which is unique identifier and a location for the gymsite.

# Implementation:

Here is an over view of the concept’s implementation:

## Object-Oriented Principles:

Here is an overview of the oop concepts used in this application

### Encapsulation:

All the classes have private attributes and public methods to access the attributes. This ensures that the object is protected from the unwanted modifications

### Inheritance:

The Trainer class serves as a base class for derived classes FitnessTrainer, YogaTrainer and MartialArtsTrainer. It allows the trainer class to implement all the redundant code at once and not to be used again and again in derived classes.

### Polymorphism:

Interfaces IGymManager and IGymUser and abstract class implements polymorphism. This allows different type of trainers to be treated as Instances of trainer class.

### Abstraction:

Interfaces and abstract class trainer implement abstraction in this application. The IGymManager and IGymUser interfaces ensures the Admin and user functionalities reapectively.

# Text Menu:

This console application includes a text-based menu system for interacting with the functionalities provided by IGymManager and IGymUser. This menu allows users to follow through the different options available to Admins and Gym Members. The menu is designed for users to interact efficiently with clear prompts and error handling to guide users through the available functionalities.

# System Functionalities:

## Personal Trainers Management:

This application allows the management of different types of personal trainers including fitness trainers, yoga trainers and martial arts trainers. Each trainer type inherits from the base class (Trainer) and ensures a consistent interface and methods across different trainer types. Trainers can be added, deleted and listed based on various sorting criteria’s such as alphabetically ordered and years of experience.

## Training Sessions Management:

This application allows the management of different session for users. Training sessions involve a member, trainer, timeslot and gym site. The application ensures that trainers are not double booked and sessions can be booked and canceled without issues. The Session is checked for trainer sessionID, timeslot and gymsite to ensure that all the information is complete and nothing is missing or irrelevant information is not added in it.

# Gym System Admin and User Functionalities:

Here is an implementation of the IGymManager and IGymUser interfaes:

## IGymManager Interface:

Admins can add new gym sites, add and delete personal trainers, list trainers alphabetically or by experience and generate session booking reports.

## IGymUser Interface:

Gym Members can view listed available trainers, view trainers by type and timeslot. They can also book sessions and cancel sessions. This application provides a user friendly interface so that it allows user to select the option and use application easily.

## Practical Implementation

### Add Gym Site:

Admins can add new gym sites by providing a site ID and location. Multiple gym sites can be added but if same siteID is given it cannot add new gymsite because the gym site id uniquely identifies a gym location across an area.

### Add Trainer:

Admins can add new trainers by providing registration number, name, experience, certifications, styles and session prices. There are 3 types of trainers admin can add any of its choice and registration number must be unique for each either another trainer with same registration number cannot be added as each number identifies a unique trainer

### Delete Trainer:

Admins can delete trainers by their registration number. If not found it returns a message showing that trainer data not exits.

### View Trainers:

Trainers can be viewed alphabetically or by experience. It provides admins and users with a tabular view to the trainer information.

### Generate Session Bookings:

Admins can generate session booking reports for a specific day which makes it easier for the admin to maintain record of bookings.

### Book and Cancel Sessions:

Members can book and cancel sessions efficiently. Each session has a unique ID so it makes it easier for the user to cancel a booked session and session with same id cannot exist.

### Conclusion:

The Gym Management System is a C# .net console application for managing gyms across different locations. It's design is too basic that make it easy to maintain and expand. It makes it easier for the gyms to manage their records and sessions for a day.