# **Name:**

# Part1 (Question-Answer)

* What is Dependency Injection, why to use it and what type of Dependency injection exist in .net?

Benefits of Dependency Injection:

Modularity: Components are decoupled, making it easier to replace or update individual parts of the application without affecting others.

Testability: Dependencies can be easily mocked or replaced during unit testing, making it simpler to write isolated and meaningful tests.

Reusability: Components with well-defined dependencies can be reused across different parts of the application.

Flexibility: It becomes easier to change or upgrade libraries, as you can replace dependencies without rewriting large portions of code.

Maintainability: The codebase becomes more maintainable and understandable, as dependencies are explicitly defined and injected.

In .NET, there are three common types of dependency injection:

1-Constructor Injection:

2-Method Injection

3-Property Injection

* What is Auto mapper, can we use it for data retrieval from database?

AutoMapper is a popular library in the .NET ecosystem that simplifies the process of mapping data between different types. It allows you to define mapping relationships between classes, and then it automates the process of copying data from one object (source) to another (destination), based on those mappings. AutoMapper helps reduce the manual and repetitive code that you would otherwise write to map properties between objects.

You can use AutoMapper for a variety of scenarios, including:

DTO Mapping: Mapping between Data Transfer Objects (DTOs) and domain/entity models. This is commonly used to transfer data between your application layers, such as from your database models to your API response objects.

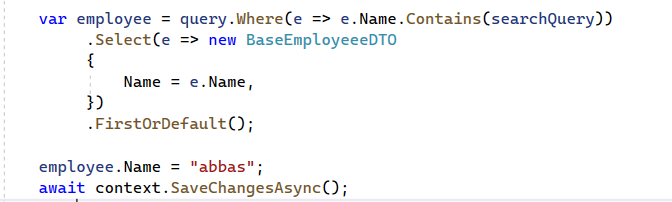
ViewModels: In ASP.NET MVC and other UI frameworks, AutoMapper can help map between domain models and view models, ensuring that only the necessary data is sent to the UI.

Projection: AutoMapper can be used to project specific properties from an object into a new object. This is particularly useful for optimizing database queries by retrieving only the necessary fields.

Entity to DTO Mapping: It's commonly used to map entity models to DTOs, which can be beneficial when you want to expose specific data to clients while keeping the entity models separated from the API.

Configuration: AutoMapper also allows you to define custom mapping configurations for more complex scenarios, and it can handle conventions-based mapping without explicit configuration.

* List 3 mistakes or best practices for this query?



1. **Point 1**: Modifying Object Before Saving Changes
2. **Point 2**: Using Contains without Case-Insensitive Comparison
3. **Point 3**: Changing Entity Property Without Updating Database

# Part2: Problem to solve

Let’s say we need to implement an order management feature for the admin panel, the administrator can list orders, edit order products and add discounts. Answer the following question:

1. Write the properties which should be in the table ***OrderProducts*** which is the table to create a many-to-many relation between the order and products.

a- OrderProductId: This is typically an auto-generated primary key for the "OrderProducts" table.

b- OrderId: A foreign key referencing the "Order" table, indicating which order the product is associated with.

c- ProductId: A foreign key referencing the "Product" table, indicating which product is included in the order.

d- Quantity: The quantity of the specific product included in the order.

e- TotalPrice: The total price of the product in the order (considering quantity and any discounts applied).

1. The order has a unique order number, the administrator says that the listing orders process is slow (orders are sorted by creation time Descending), how can we improve? List three must be implemented practices to speed up the listing process
   1. Point 1: Proper Indexing
   2. Point 2: Pagination
   3. Point 3: Caching

1. How can we track changes of the order like status?

 Audit Log or History Table

Database Triggers

 Using Entity Framework Core Auditing

External Logging or Monitoring Tools

1. How can we prevent concurrent changes on the order record?

Optimistic Concurrency Control

Pessimistic Concurrency Control

Row-Level Database Locking

**Good Luck**