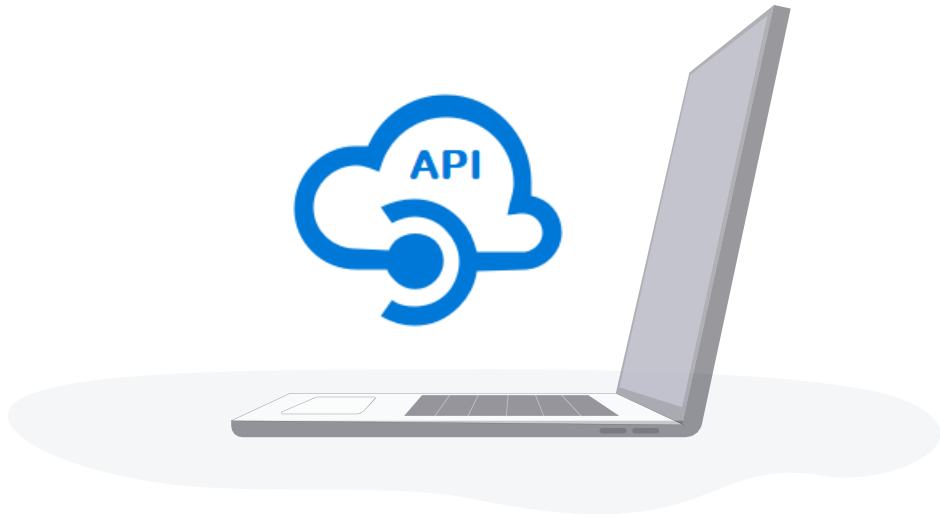
Harmony IT Solution

Web Application Programming Interface (API)

Tahaluf Training Center 2022

Harmony IT Solution

1. Overview of Onion Architecture
2. Layers of Onion Architecture
3. Benefits of Onion Architecture
4. Overview Of DB Context
5. Create DB Context
6. Create Domain Layer (Database First)



Overview of Onion Architecture

The majority of traditional architectures have inherent flaws with tight coupling and separation of concerns.

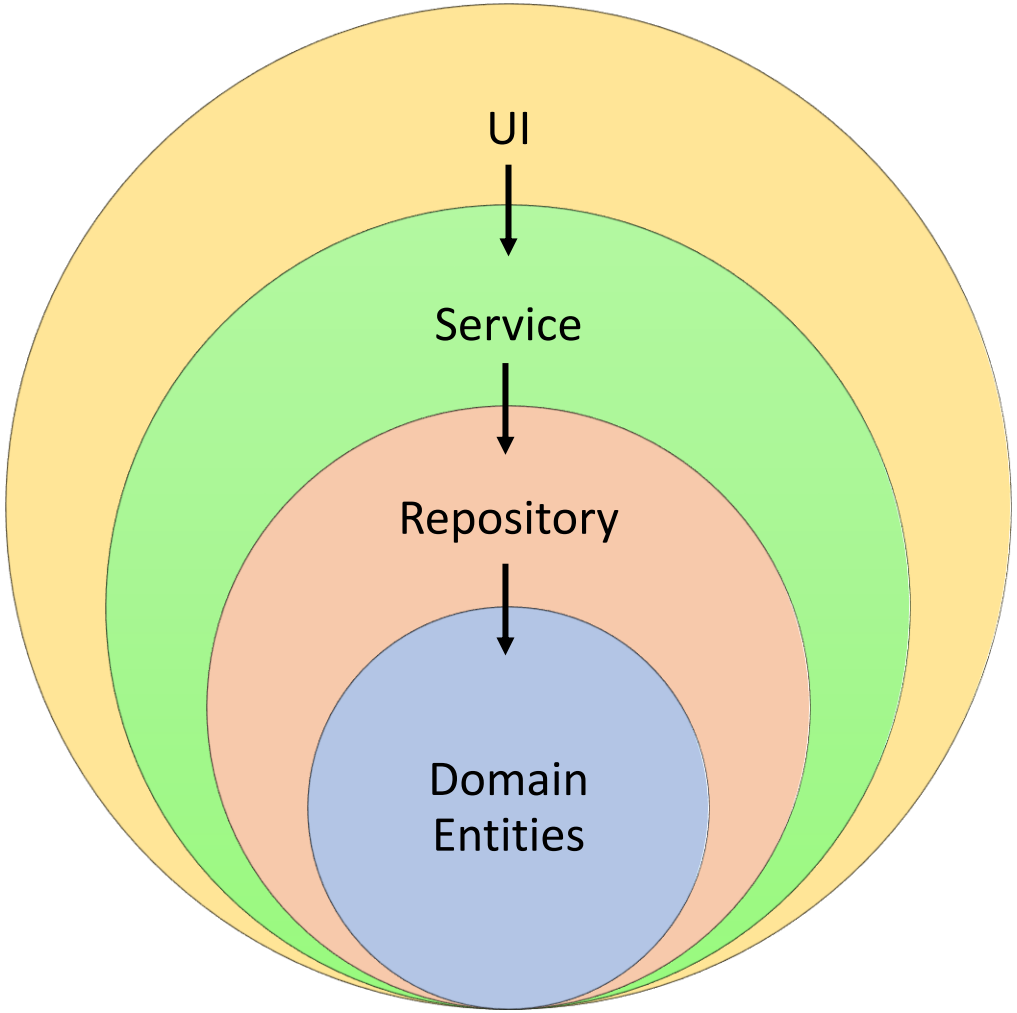
Jeffrey Palermo proposed the Onion Architecture to give a better approach to build applications in terms of testability, maintainability, and reliability.

Onion Architecture was created to solve the issues that 3-tier architectures face, as well as to give a solution to common challenges.

Interfaces are used by onion architecture layers to communicate with one another.

Harmony IT Solution

Layers of Onion Architecture



Domain Layer

The domain layer, which represents the business and behavior objects, is located in the heart of the Onion Architecture. The goal is to have this core contain all of your domain objects. It's where you'll find all of your application's domain objects.

Domain Layer

You could have domain interfaces in addition to domain objects. There are no dependencies between these domain objects. Without any heavy code or dependencies, domain objects are likewise flat, as they should be.

Repository Layer

This layer creates an abstraction between an application's domain entities and its business logic. We normally include Interfaces at this layer that provide object saving and retrieving functionality, usually through the use of a database. The data access pattern, which is a more loosely coupled approach to data access, is part of this layer.

Repository Layer

We also build a generic repository and implement queries to extract data from the source, map data from the data source to a business entity, and persist business entity updates to the data source.

Service Layer

Add, Save, Edit, and Delete some common activities available through the Service layer. This layer also acts as a link between the UI and repository layers. The entity's business logic might potentially be stored in the Service layer. Service interfaces are maintained separate from their implementation on this layer, ensuring loose coupling and separation of concerns.

UI

It's the top layer, and it's where things like UI and tests are kept. It represents the Web API for the Web application. The dependency injection principle is used in this layer, allowing the application to design a loosely linked structure and communicate with the internal layer via interfaces.

Benefits of Onion Architecture

- Interfaces link the layers of the onion architecture.
- A domain model forms the base for application architecture.
- All external dependencies, like database access and service calls, are represented in external layers.
- There are no external layers that are dependent on the Internal layer.

Benefits of Onion Architecture

- The couplings are at the center.
- Architecture that is flexible, sustainable, and portable.
- Because the application core is independent of everything, it can be easily tested.

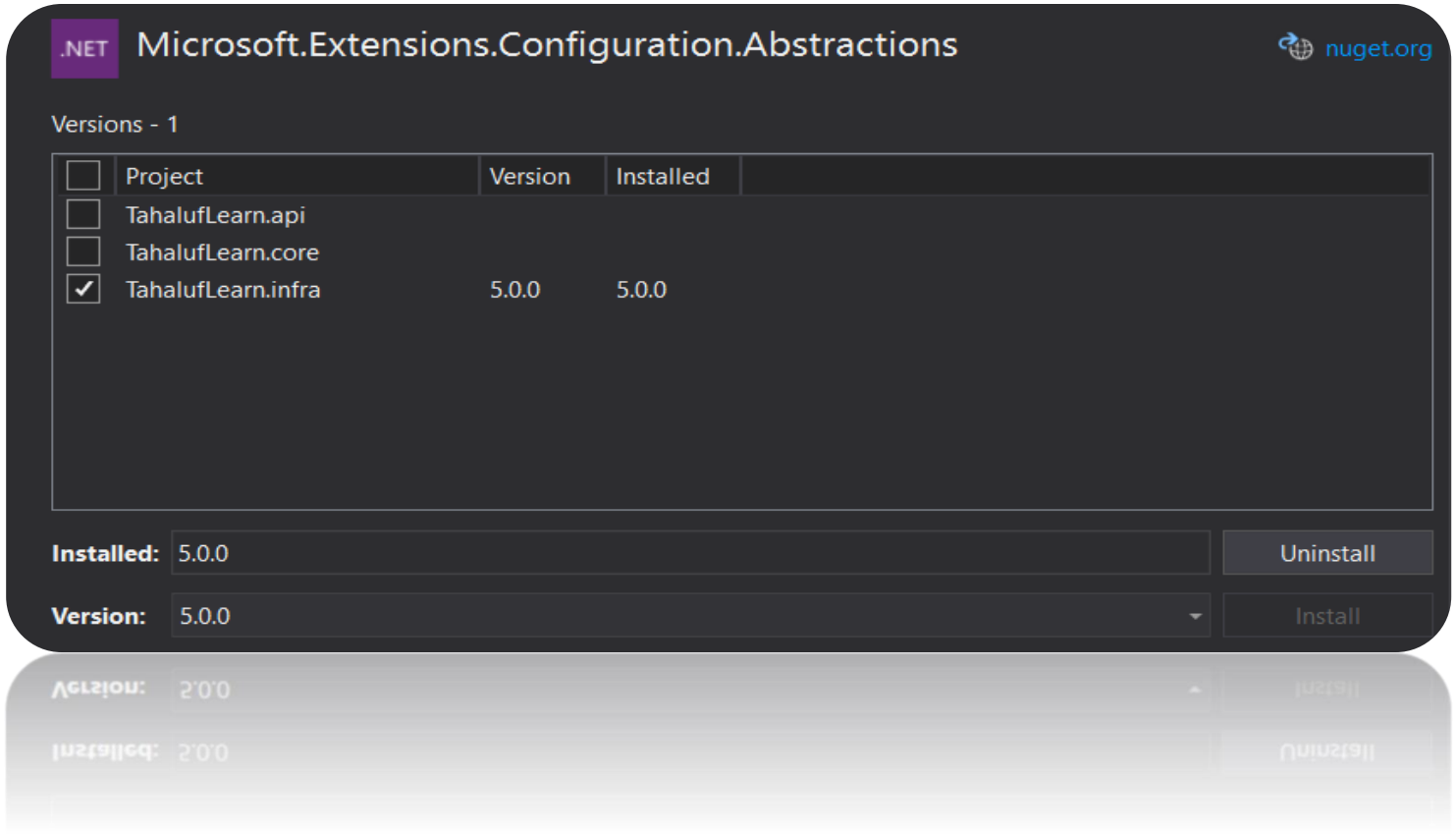
Overview of DB Context

A DbContext instance is a session with the database used to retrieve and store instances of your entities.

Create DB Context

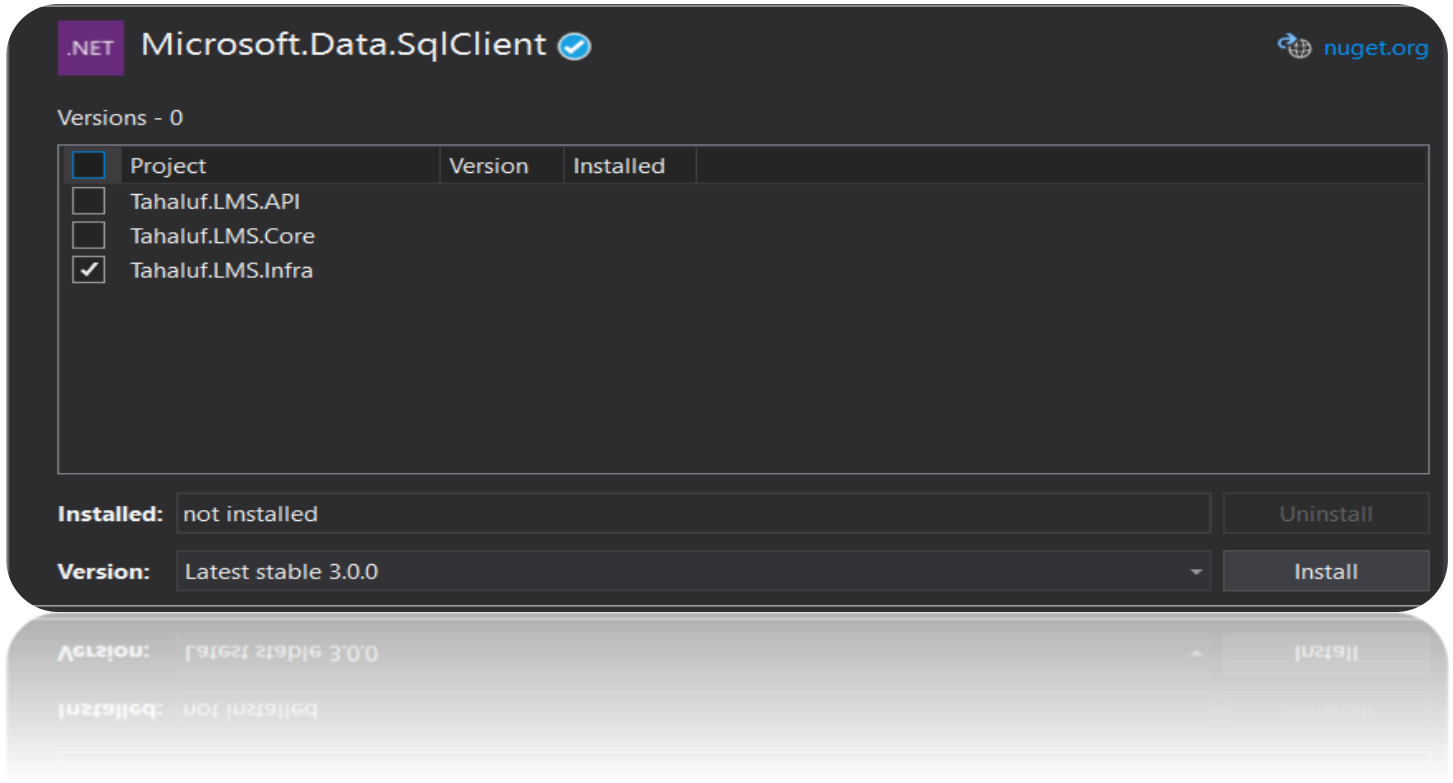
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install Microsoft.Extensions.Configuration.Abstractions



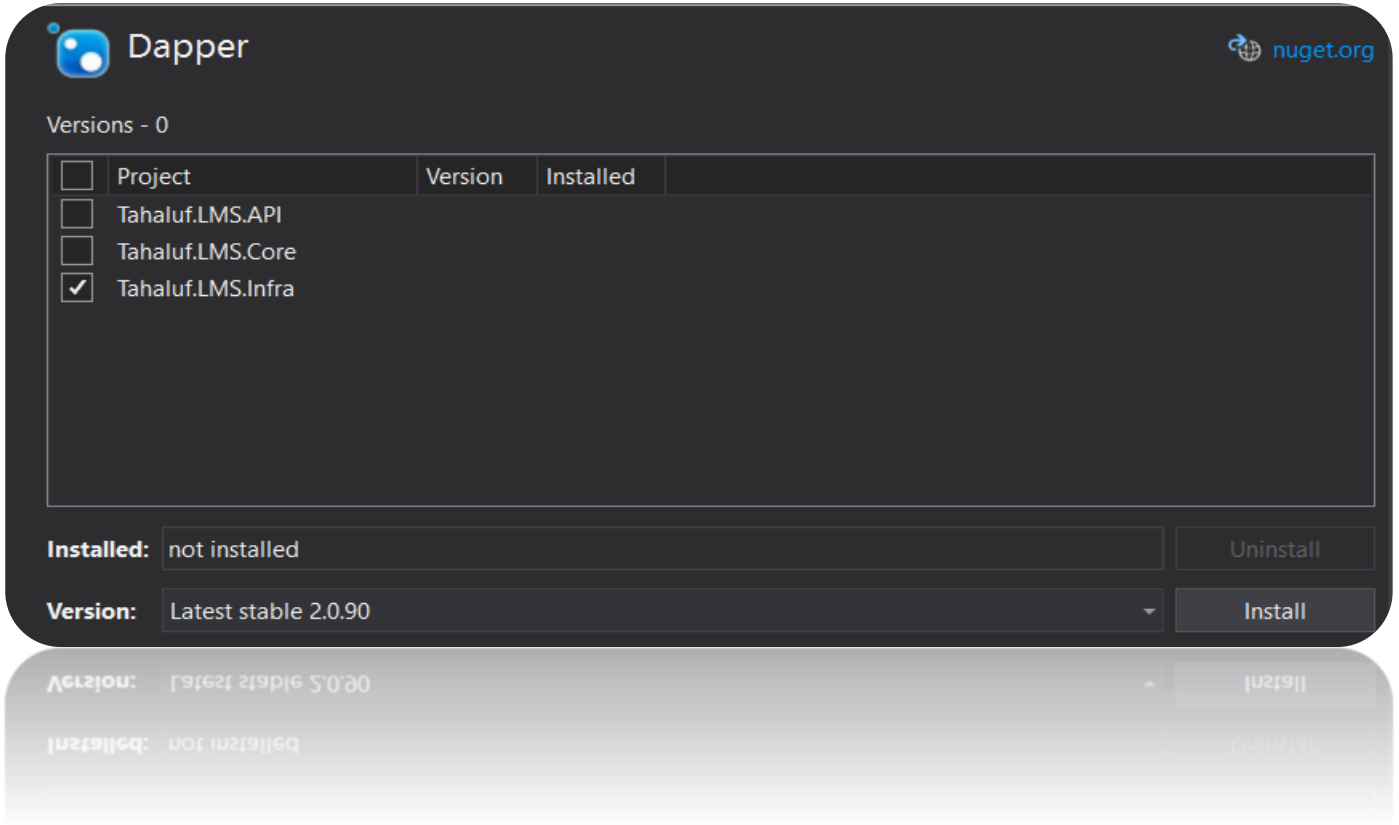
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install Microsoft.Data.SqlClient.



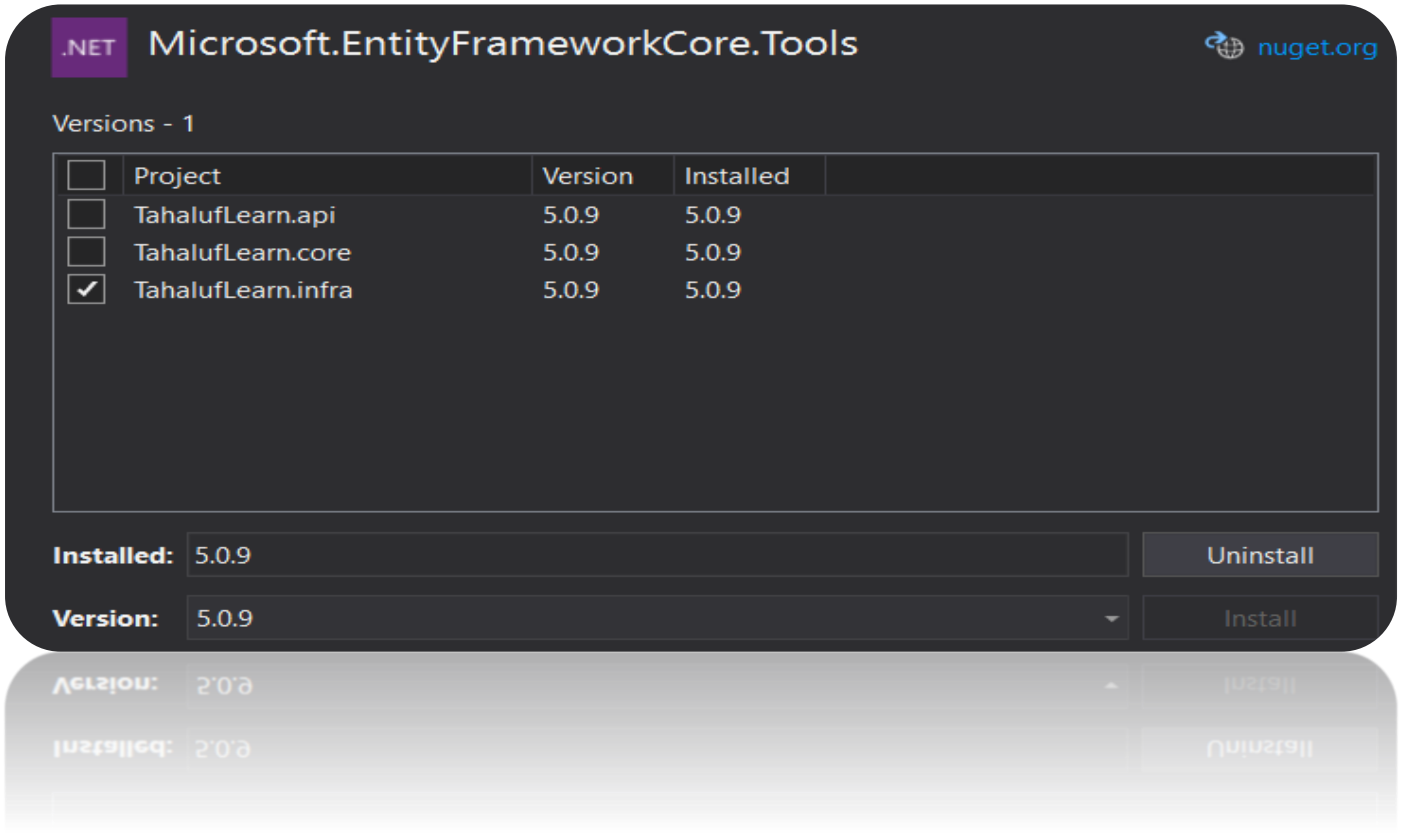
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install Dapper.



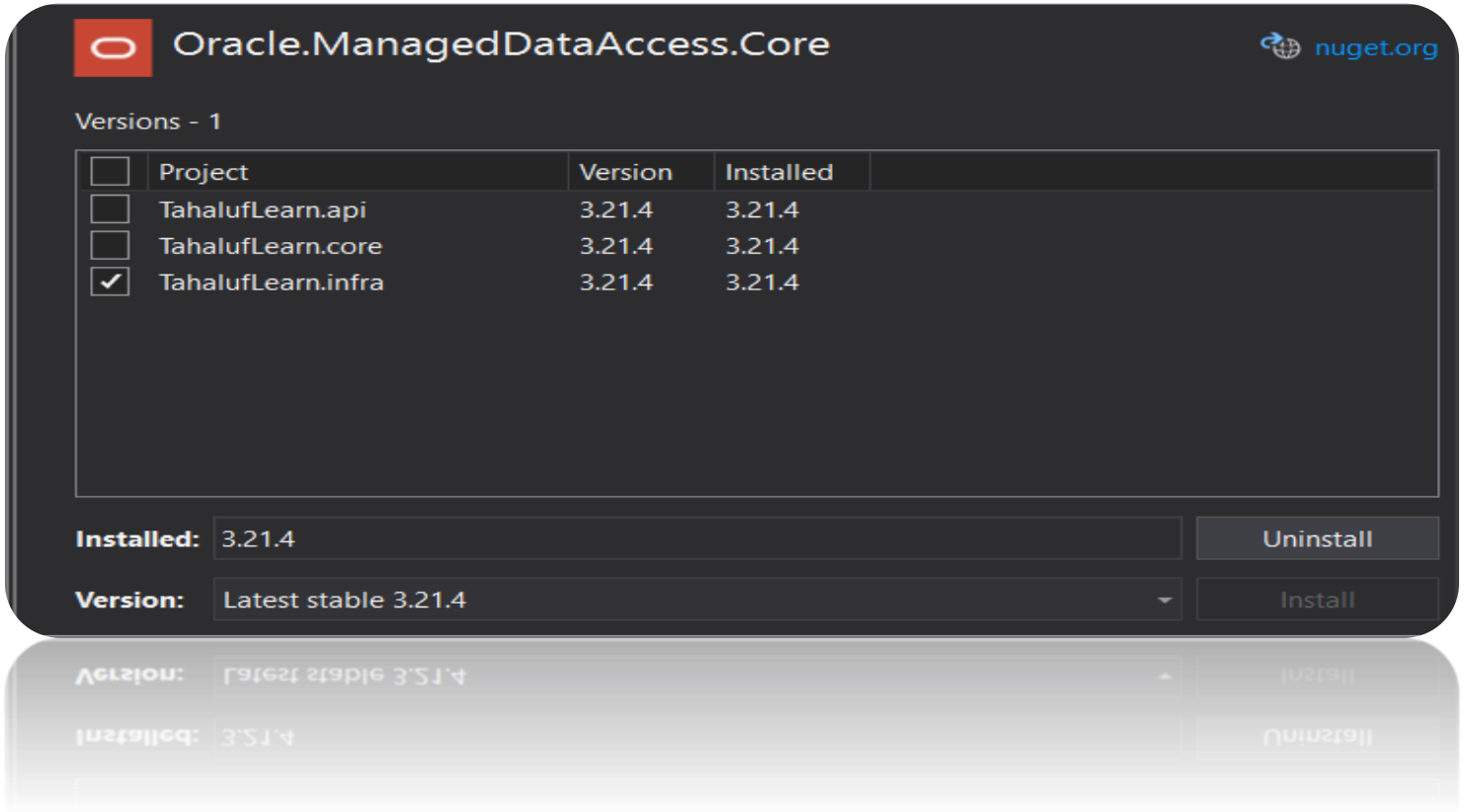
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Microsoft.EntityFrameworkCore.Tools



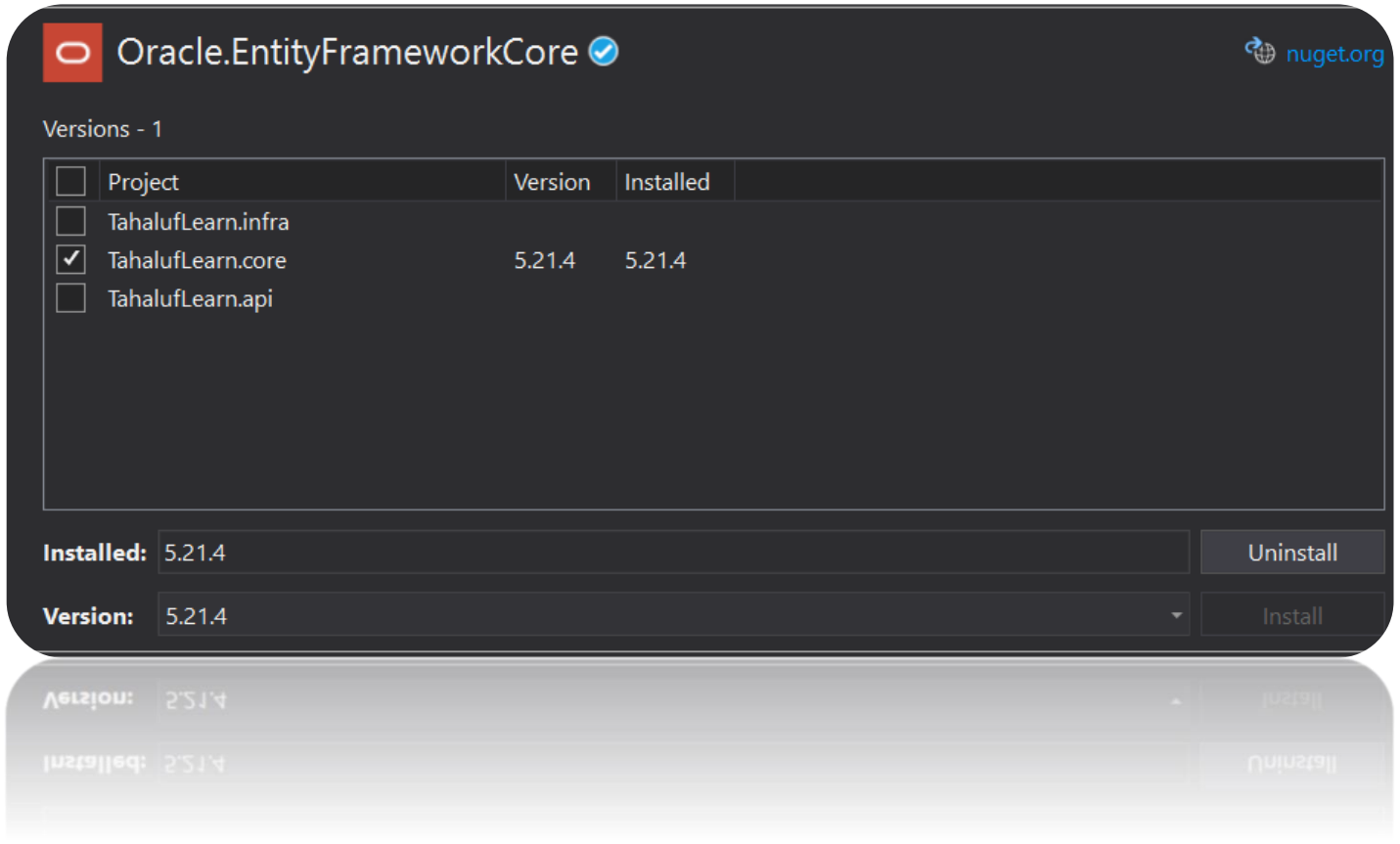
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Oracle.ManagedDataAccess.Core



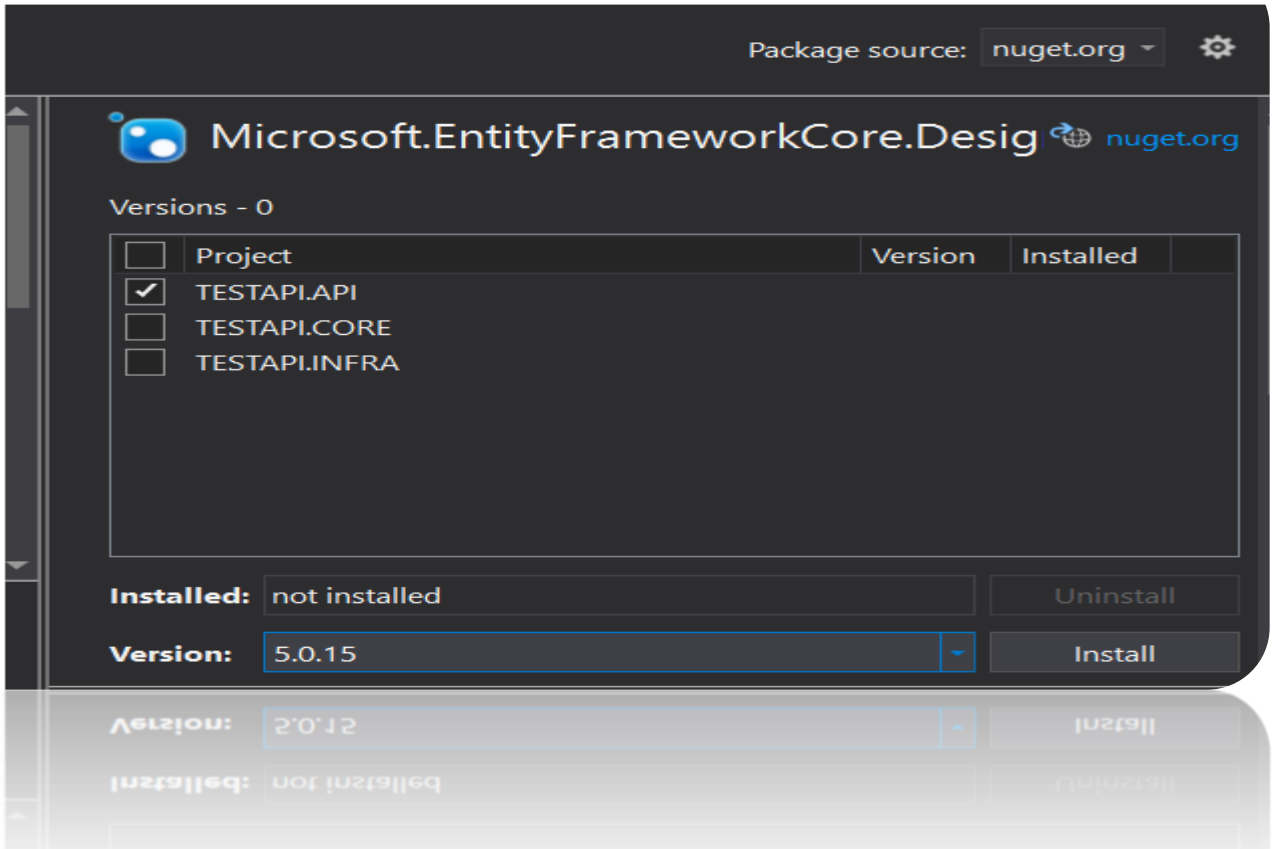
Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Oracle. EntityFrameworkCore



Install Packages

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Microsoft.EntityFrameworkCore.Design



Database Connection

Write the following Connection String in app Setting.json:

```
"ConnectionStrings": {
    "DBConnectionString": "Data Source=(DESCRIPTION =(ADDRESS
= (PROTOCOL = TCP)(HOST = 94.56.229.181)(PORT =
3488))(CONNECT_DATA =(SERVER = DEDICATED)(SERVICE_NAME =
traindb))); User Id=JOR17_User92;PASSWORD=Test321;Persist
Security Info=True;"
  },
```

Create Common Folder

Right Click on TahalufLearn.Infra => Add => New Folder => Common.

Right Click on TahalufLearn.core => Add => New Folder => Common.

Create DbContext Class and Interface

Right Click on Common in TahalufLearn.Core => Add => Class => Choose Interface => IDbContext.

Right Click on Common in TahalufLearn.Infra => Add => Class => DbContext.

Note: Set Class and Interface public.

IDbContext Code:

```
public interface IDbContext
{
        public interface IDbContext
        {
            DbConnection Connection { get; }
        }

}
```

DbContext Code:

```
public class DbContext: IDbContext
{
        private DbConnection _connection;
        private readonly IConfiguration _configuration;

        public DbContext(IConfiguration configuration)
        {
            _configuration = configuration;
        }
```

DbContext Code:

```
public DbConnection Connection
{
      get
          {

            if (_connection == null)
              {
                _connection = new OracleConnection
                (_configuration["ConnectionStrings:
                DBConnectionString"]);
                _connection.Open();
              }
```

DbContext Code:

```
else if (_connection.State != ConnectionState.Open)
   {
       _connection.Open();
    }

    return _connection;

}
```

Add Services in Startup

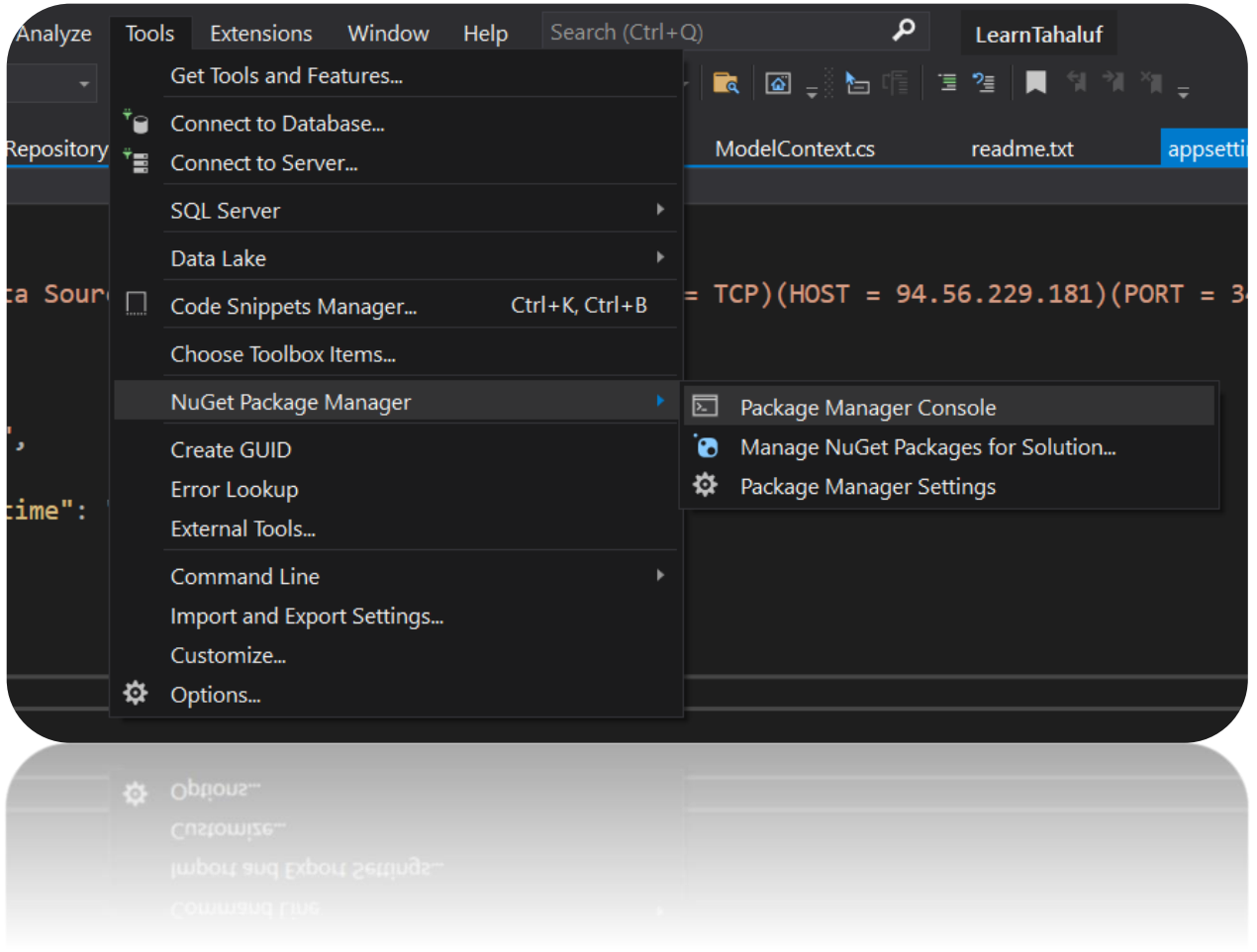
Write the following code in Configure services:

```
services.AddScoped<IDbContext, DbContext>();
```

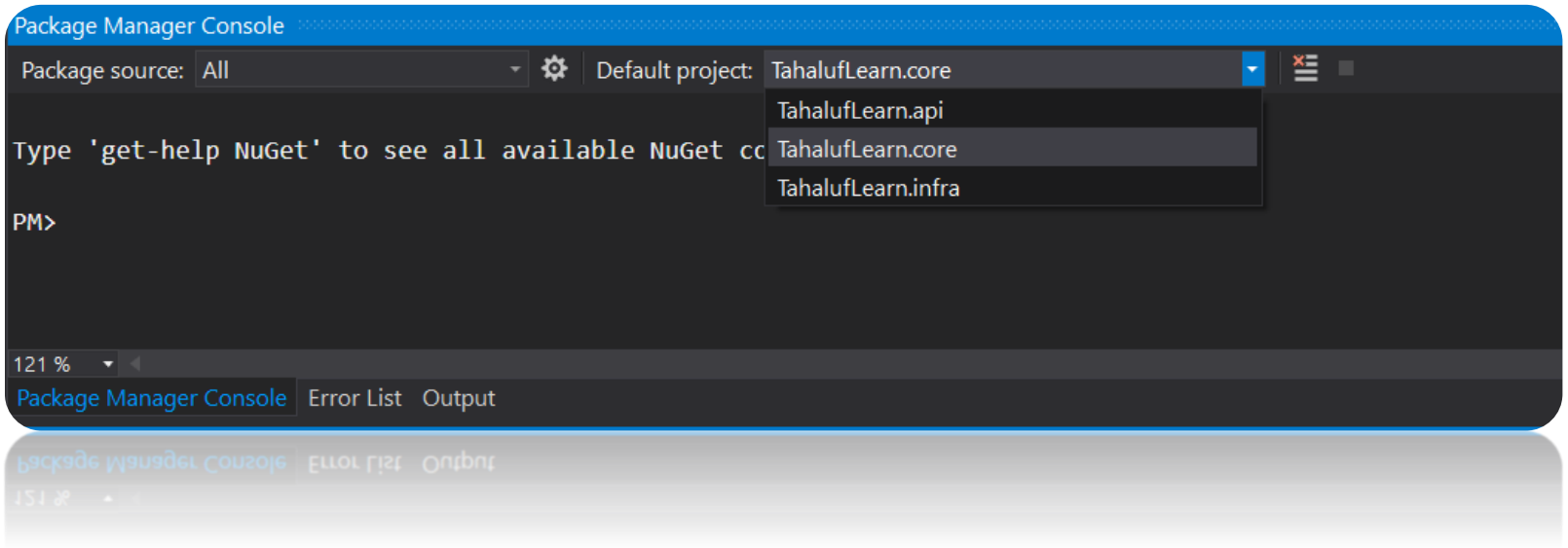
Harmony IT Solution

Create Domain Layer (Database First)

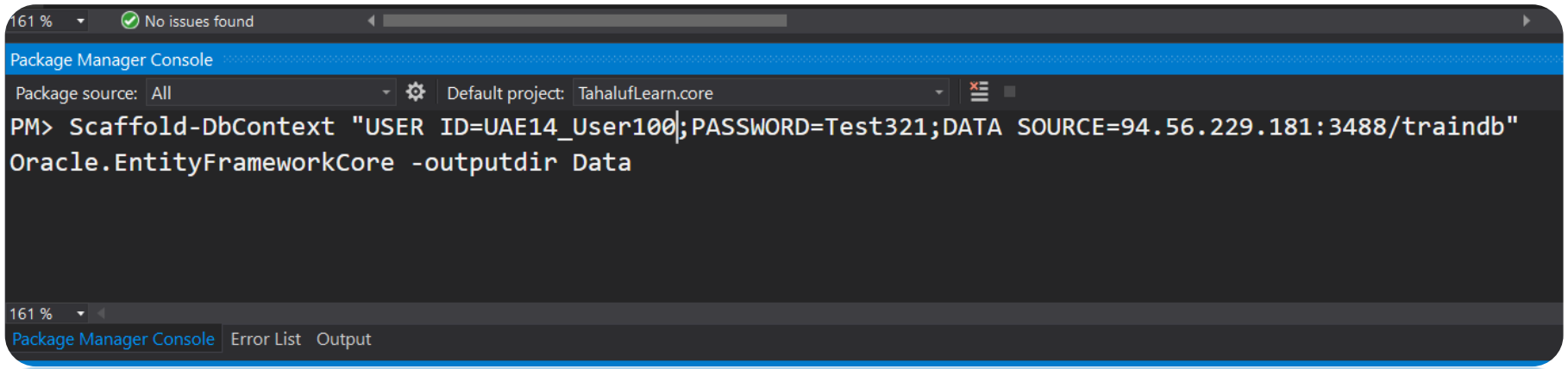
Tools => NuGet Package Manager => Package Manager Console.



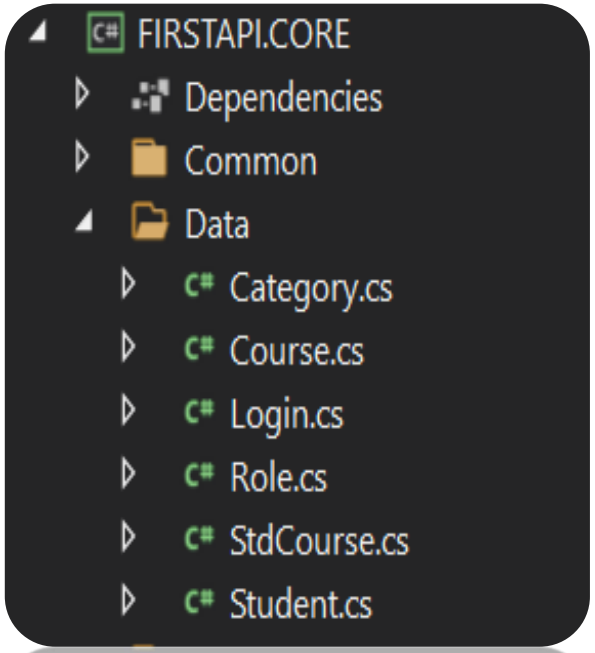
Package Manager Console => Default Project => TahalufLearn.Core



Scaffold-DbContext "User Id=JOR17_User92;PASSWORD=Test321;DATA SOURCE=94.56.229.181:3488/traindb" Oracle.EntityFrameworkCore -outputdir Data



Harmony IT Solution




```cs
public partial class Course
{
    0 references
    public Course()
    {
        Stdcourses = new HashSet<Stdcourse>();
    }

    1 reference
    public decimal Courseid { get; set; }
    1 reference
    public string Coursename { get; set; }
    2 references
    public decimal? Categoreyid { get; set; }

    1 reference
    public virtual Categorey Categorey { get; set; }
    2 references
    public virtual ICollection<Stdcourse> Stdcourses { get; set; }
}
```

```cs
public partial class Category
{
    0 references
    public Category()
    {
        Courses = new HashSet<Course>();
    }

    0 references
    public decimal Categoryid { get; set; }
    0 references
    public string Categoryname { get; set; }

    1 reference
    public virtual ICollection<Course> Courses { get; set; }
}
```

```cs
0 references
public partial class Login
{
    1 reference
    public decimal Loginid { get; set; }
    2 references
    public string Username { get; set; }
    1 reference
    public string Password { get; set; }
    1 reference
    public decimal? Isactive { get; set; }
    2 references
    public decimal? Roleid { get; set; }
    2 references
    public decimal Studentid { get; set; }

    1 reference
    public virtual Role Role { get; set; }
    1 reference
    public virtual Stduent Student { get; set; }
}
```

[1]. https://www.codeguru.com/csharp/understanding-onion-architecture/#:~:text=Onion%20Architecture%20is%20based%20on,on%20the%20actual%20domain%20models

[2]. https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-5.0