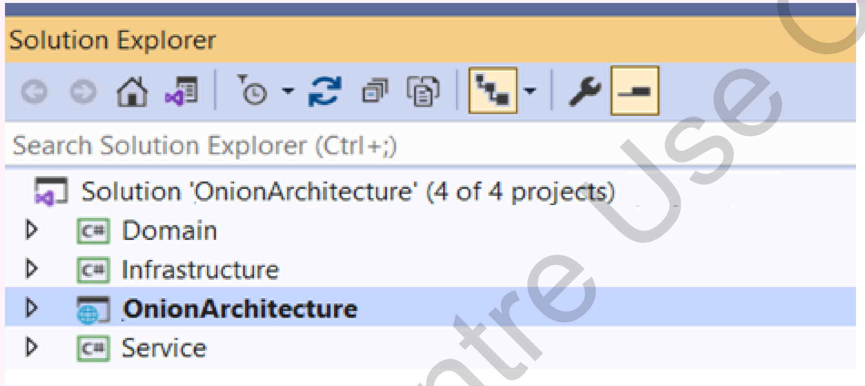
Session 10

Onion Architecture in ASP.NET Core - II

Session Overview

- Explain project structure for Onion Architecture
- Describe the process of implementation of the four projects – Domain layer, Infrastructure layer, Service layer, and Onion Architecture Web API

Project Structure of Onion Architecture

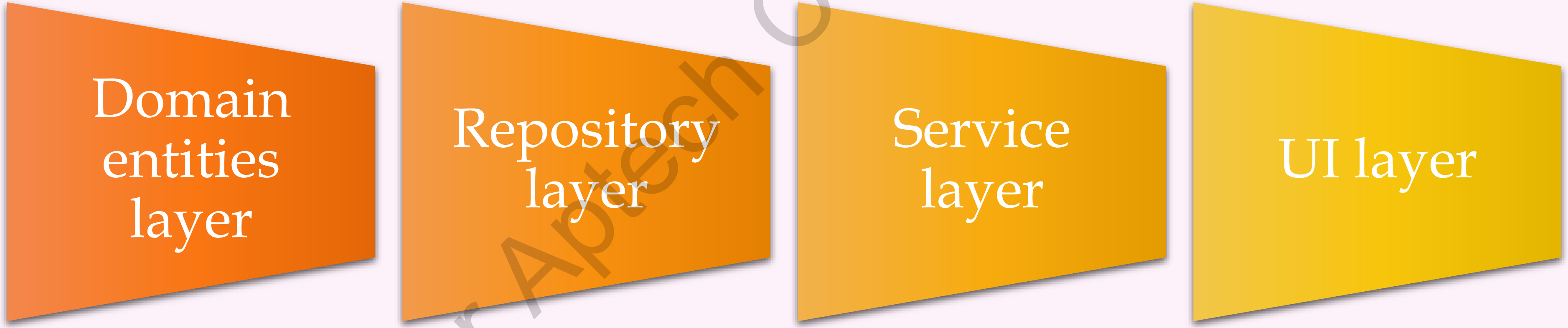


Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'OnionArchitecture' (4 of 4 projects)
- Domain
- Infrastructure
- OnionArchitecture
- Service

Figure 10.1: Layers of Onion Architecture



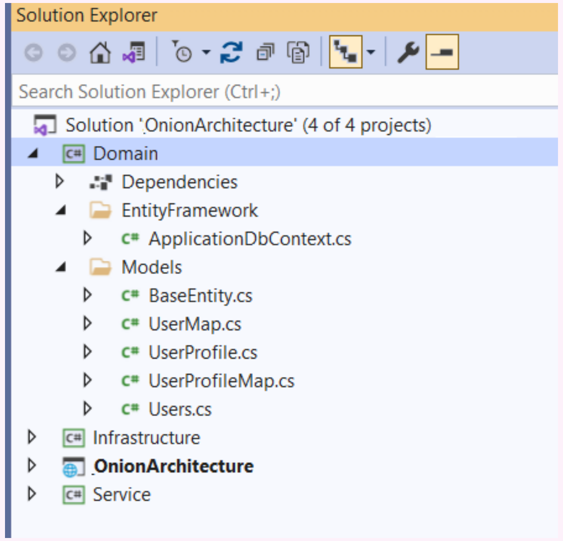
Domain entities layer

Repository layer

Service layer

UI layer

Domain Entities Layer (1-2)

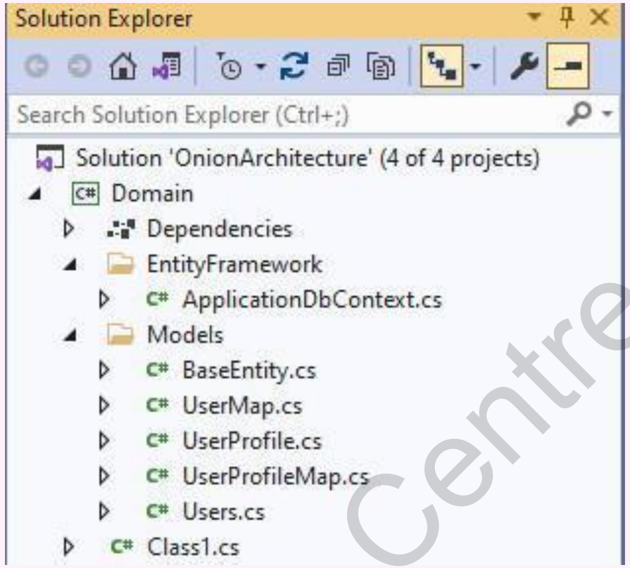


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'OnionArchitecture' (4 of 4 projects)
  Domain
    Dependencies
    EntityFramework
      ApplicationDbContext.cs
    Models
      BaseEntity.cs
      UserMap.cs
      UserProfile.cs
      UserProfileMap.cs
      Users.cs
  Infrastructure
  OnionArchitecture
  Service
```

Figure 10.2: Domain Entity Layer

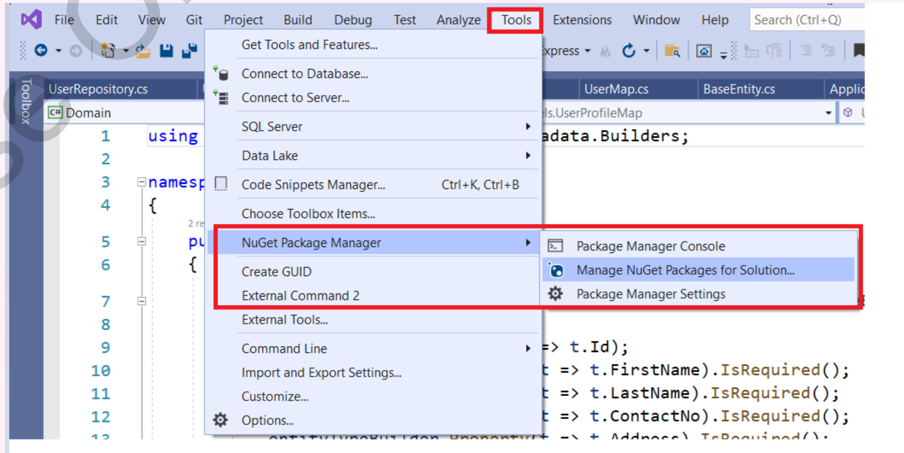


Solution Explorer

Search Solution Explorer (Ctrl+;)

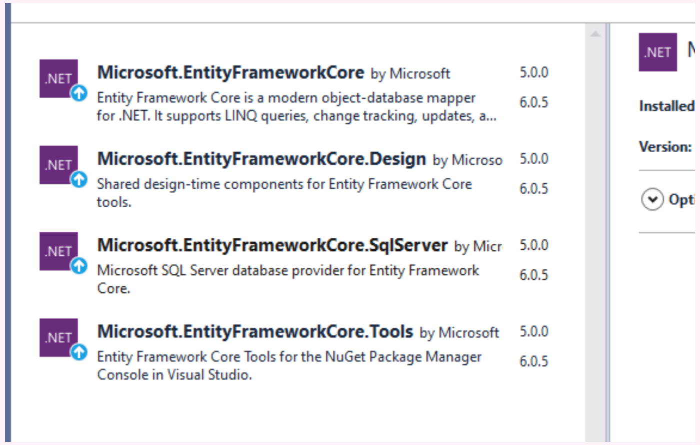
```
Solution 'OnionArchitecture' (4 of 4 projects)
  Domain
    Dependencies
    EntityFramework
      ApplicationDbContext.cs
    Models
      BaseEntity.cs
      UserMap.cs
      UserProfile.cs
      UserProfileMap.cs
      Users.cs
    Class1.cs
```

Figure 10.3: Model and EntityFramework Folders



```
using                  adata.Builders;
namesp
{
    pu
    {
                       => t.Id);
                       t => t.FirstName).IsRequired();
                       t => t.LastName).IsRequired();
                       t => t.ContactNo).IsRequired();
```

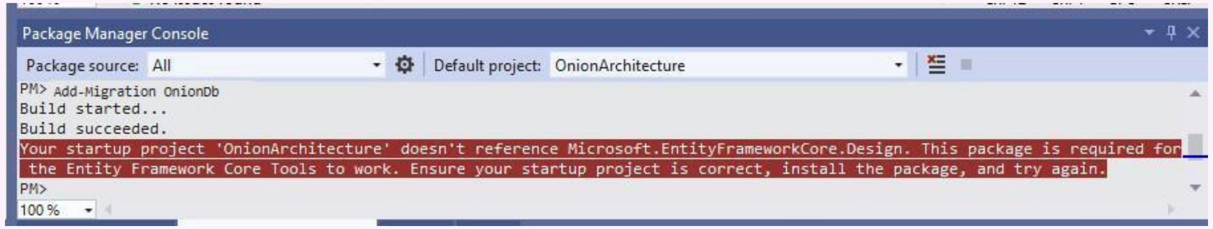
Figure 10.4: NuGet Package Manager



Package	Versions
Microsoft.EntityFrameworkCore by Microsoft Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, a...	5.0.0 6.0.5
Microsoft.EntityFrameworkCore.Design by Microso Shared design-time components for Entity Framework Core tools.	5.0.0 6.0.5
Microsoft.EntityFrameworkCore.SqlServer by Micr Microsoft SQL Server database provider for Entity Framework Core.	5.0.0 6.0.5
Microsoft.EntityFrameworkCore.Tools by Microsoft Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	5.0.0 6.0.5

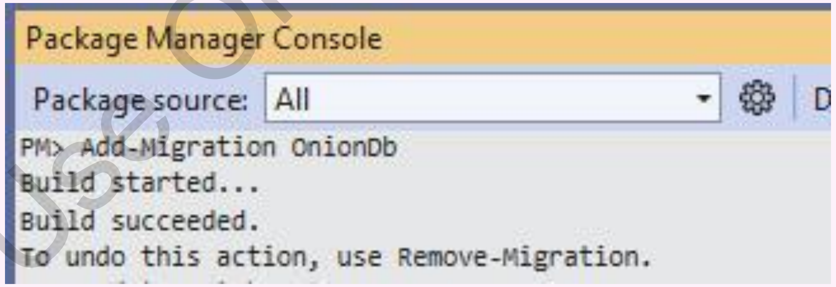
Figure 10.5: EntityFramework Projects

Domain Entities Layer (2-2)



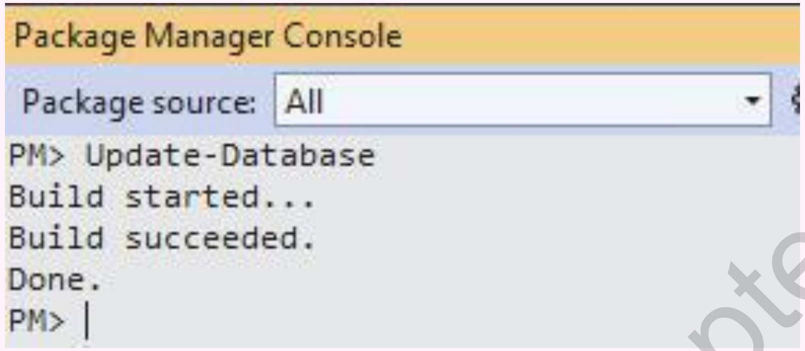
```
PM> Add-Migration OnionDb
Build started...
Build succeeded.
Your startup project 'OnionArchitecture' doesn't reference Microsoft.EntityFrameworkCore.Design. This package is required for
 the Entity Framework Core Tools to work. Ensure your startup project is correct, install the package, and try again.
PM>
```

Figure 10.6: Package Manager Console



```
PM> Add-Migration OnionDb
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
```

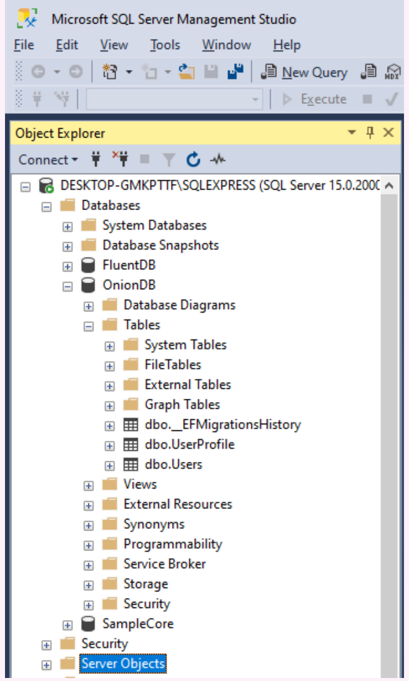
Figure 10.7: Build Successful Message



```
PM> Update-Database
Build started...
Build succeeded.
Done.
PM> |
```

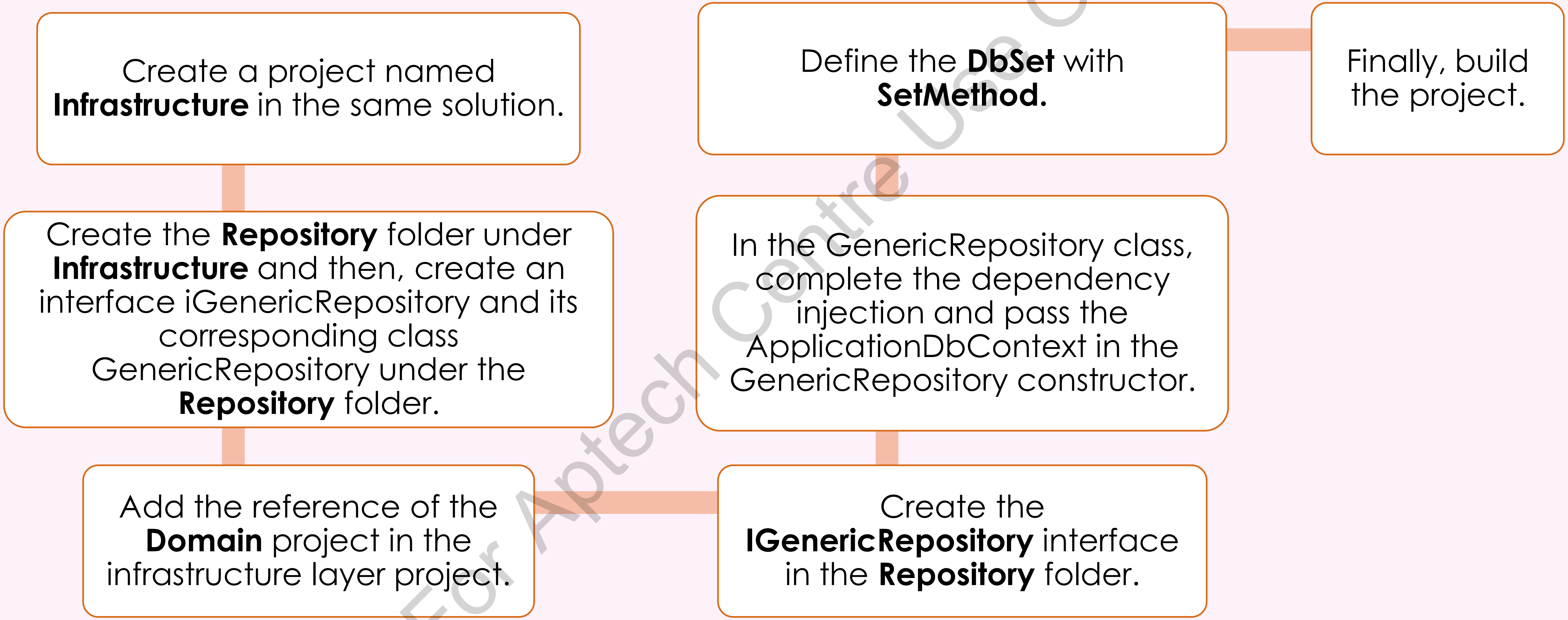
Figure 10.8: Update Database

Figure 10.9: Database and Tables



```
DESKTOP-GMKPTTF\SQLEXPRESS (SQL Server 15.0.200(
  Databases
    System Databases
    Database Snapshots
    FluentDB
    OnionDB
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.__EFMigrationsHistory
        dbo.UserProfile
        dbo.Users
      Views
      External Resources
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
    SampleCore
  Security
  Server Objects
```

Infrastructure Layer



Create a project named **Infrastructure** in the same solution.

Create the **Repository** folder under **Infrastructure** and then, create an interface iGenericRepository and its corresponding class GenericRepository under the **Repository** folder.

Add the reference of the **Domain** project in the infrastructure layer project.

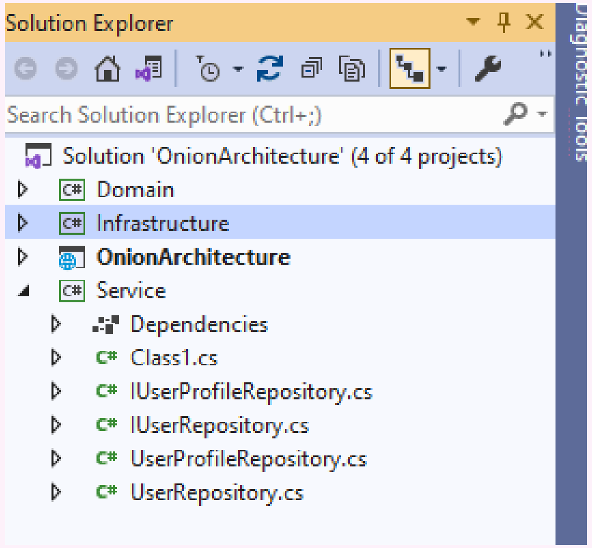
Create the **IGenericRepository** interface in the **Repository** folder.

In the GenericRepository class, complete the dependency injection and pass the ApplicationDbContext in the GenericRepository constructor.

Define the **DbSet** with **SetMethod.**

Finally, build the project.

Service Layer

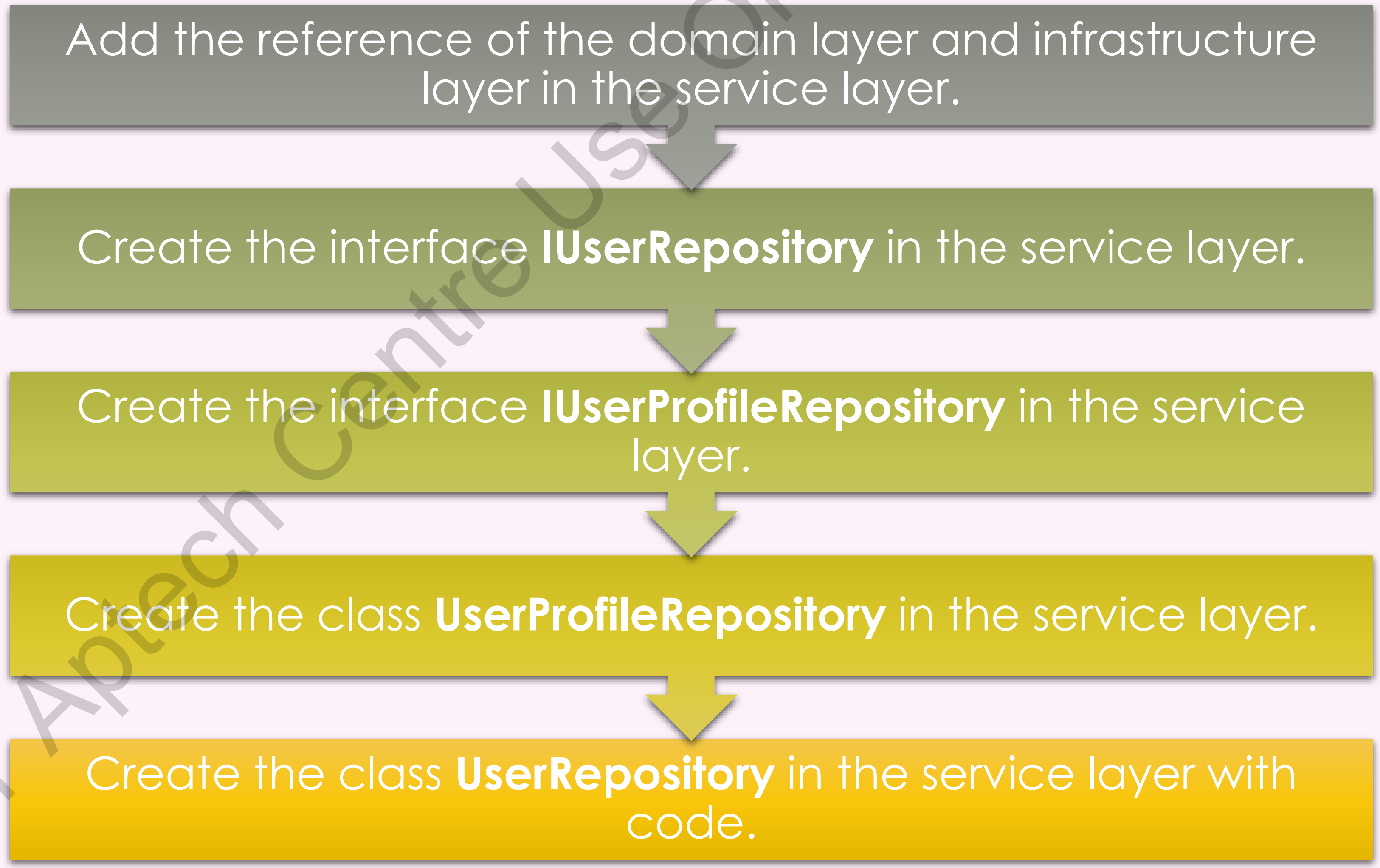


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'OnionArchitecture' (4 of 4 projects)
▷ Domain
▷ Infrastructure
▷ OnionArchitecture
◢ Service
    ▷ Dependencies
    ▷ Class1.cs
    ▷ IUserProfileRepository.cs
    ▷ IUserRepository.cs
    ▷ UserProfileRepository.cs
    ▷ UserRepository.cs
```

Figure 10.12: Service Layer



Add the reference of the domain layer and infrastructure layer in the service layer.

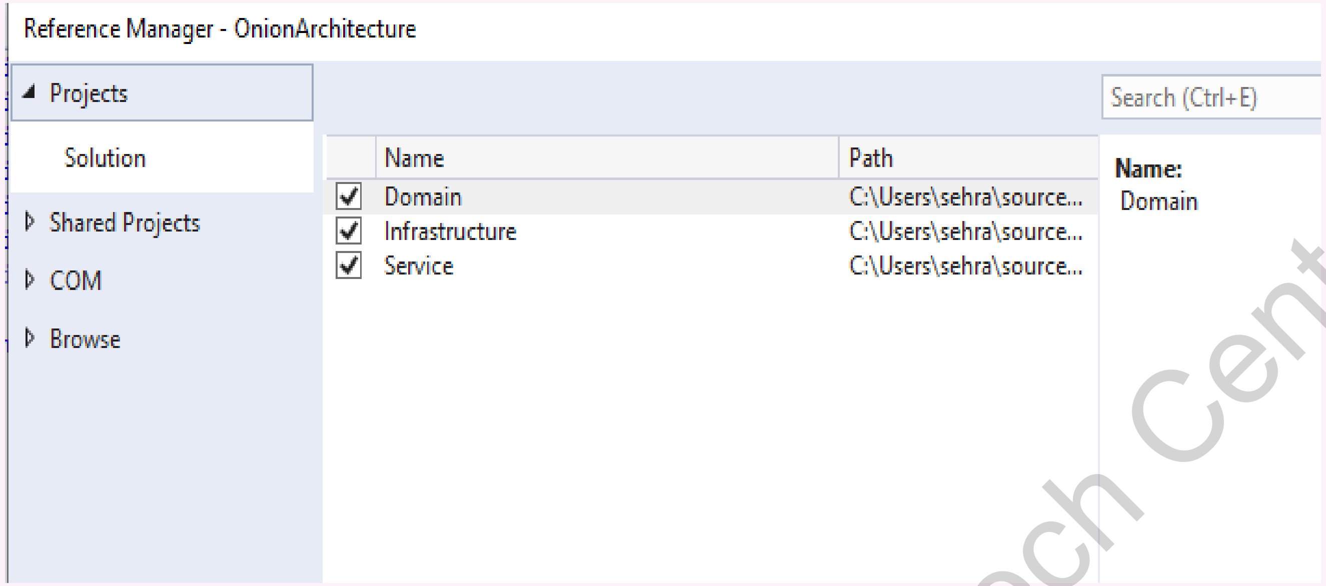
Create the interface **IUserRepository** in the service layer.

Create the interface **IUserProfileRepository** in the service layer.

Create the class **UserProfileRepository** in the service layer.

Create the class **UserRepository** in the service layer with code.

Onion Architecture Web API (1-2)



Reference Manager - OnionArchitecture

	Name	Path
☑	Domain	C:\Users\sehra\source...
☑	Infrastructure	C:\Users\sehra\source...
☑	Service	C:\Users\sehra\source...

Name: Domain

Figure 10.13: Onion Architecture

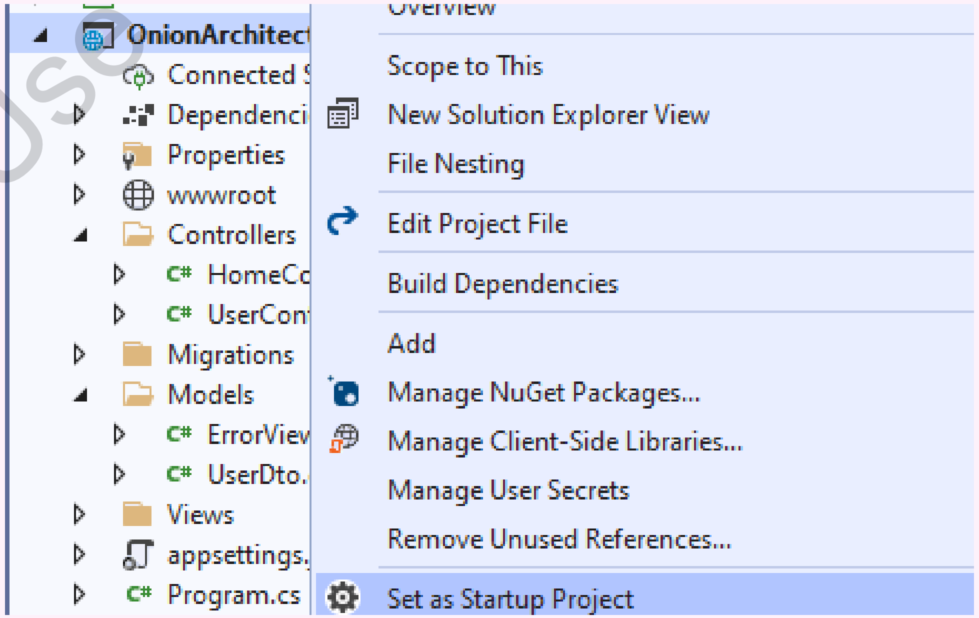
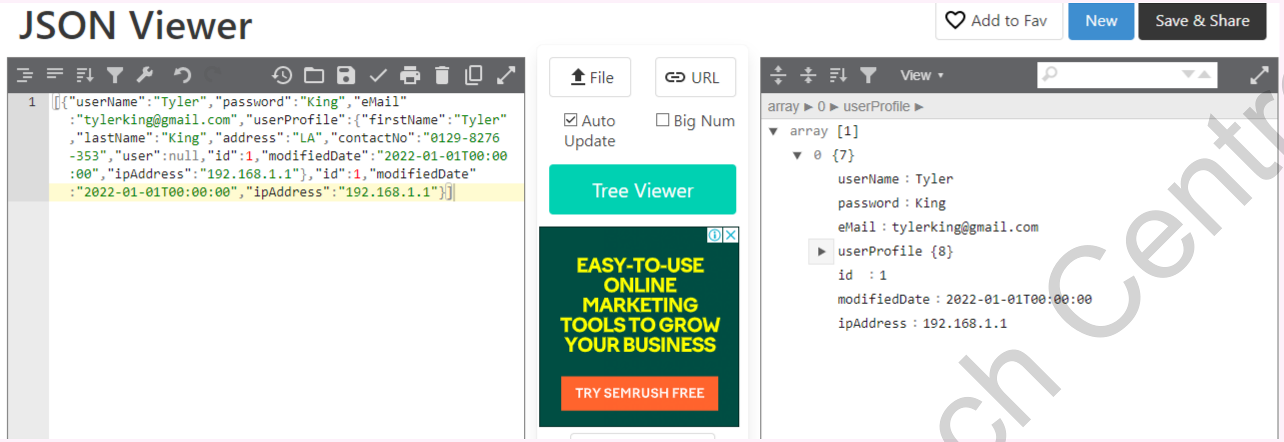


Figure 10.14: Set as Startup Project

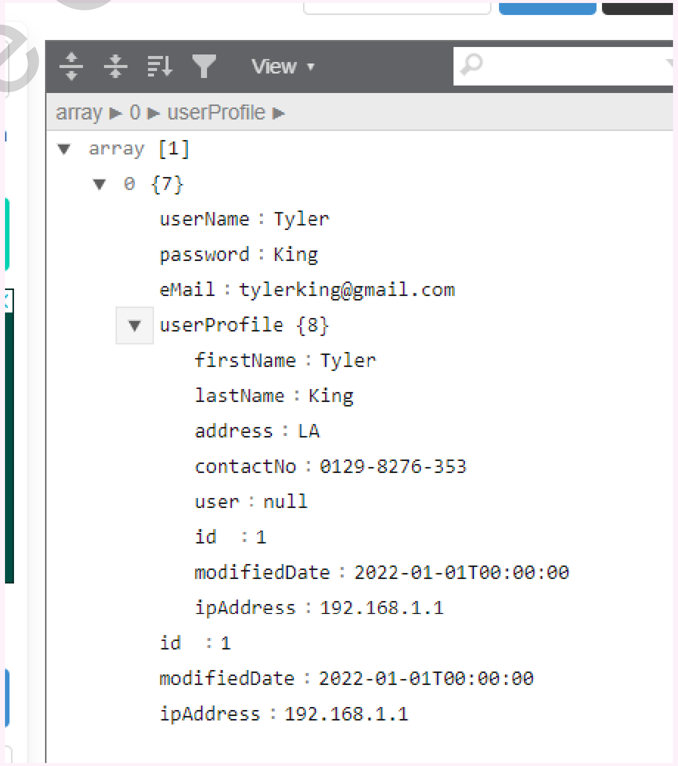
Onion Architecture Web API (2-2)



```
1  [{"userName":"Tyler","password":"King","eMail"
   :"tylerking@gmail.com","userProfile":{"firstName":"Tyler"
   ,"lastName":"King","address":"LA","contactNo":"0129-8276
   -353","user":null,"id":1,"modifiedDate":"2022-01-01T00:00
   :00","ipAddress":"192.168.1.1"},"id":1,"modifiedDate"
   :"2022-01-01T00:00:00","ipAddress":"192.168.1.1"}]
```

```
▼ array [1]
  ▼ 0 {7}
      userName : Tyler
      password : King
      eMail : tylerking@gmail.com
    ▶ userProfile {8}
      id  : 1
      modifiedDate : 2022-01-01T00:00:00
      ipAddress : 192.168.1.1
```

Figure 10.15: JSON Viewer



```
▼ array [1]
  ▼ 0 {7}
      userName : Tyler
      password : King
      eMail : tylerking@gmail.com
    ▼ userProfile {8}
        firstName : Tyler
        lastName : King
        address : LA
        contactNo : 0129-8276-353
        user : null
        id  : 1
        modifiedDate : 2022-01-01T00:00:00
        ipAddress : 192.168.1.1
      id  : 1
      modifiedDate : 2022-01-01T00:00:00
      ipAddress : 192.168.1.1
```

Figure 10.16: User Profile

Summary

- ✓ Four different layers of the Onion Architecture are domain entities layer, repository layer, service layer, and UI layer.
- ✓ For each of these layers, a corresponding project must be created to implement the Onion Architecture in an ASP.NET Core application.
- ✓ Domain entities layer contains the class library, POCO, and configuration classes. It also helps in creating database tables.
- ✓ The repository layer implements the interface for the generic repository class. It also contains the DbContext class.
- ✓ The service layer contains the business logic and user interfaces.
- ✓ The UI layer is the entry point of the program and the exterior layer of the Onion Architecture.