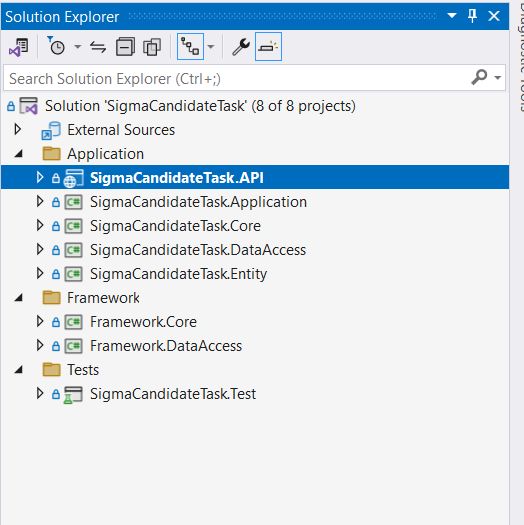
# **Technical Document && Solution Structure**

## **Solution Structure**

I have implemented Onion Architecture as the below



### **Application**

* **SigmaCandidateTask.Entity:** this layer will represent the entity.
* **SigmaCandidateTask.DataAccess :** this layer will contains :
  + Repository Implementations for the entity.
  + Migrations.
  + Entity Mapping (Fluent API).
* **SigmaCandidateTask.Application:** this layerwillcontain services or business implementations
* **SigmaCandidateTask.API:** this layerwillcontain:
  + **Controller** that represents the endpoints.
  + **Middlewares** like ExceptionHandlingMiddleware.
  + **Logs**
  + **Program.cs** that contains general configurations.
* **SigmaCandidateTask.Core:** this layerwillcontain
  + **Abstractions (**interface for Repository and services**)**
  + **AutoMapper**
  + **ViewModels**

### **Framework**

* **Framework.DataAccess :** this layerwillcontain the Implementations of generic Repository or all Framework Implementations
* **Framework. Core** this layerwillcontain the Framework **Abstractions and Models**

### **Tests**

* **SigmaCandidateTask.Test:** this layerwillcontain the Implementations Unit Test

## **Technologies and tools**

1. ASP .Net 8
2. SQL Server 2019
3. Visual Studio 2022 as IDE
4. Github as Source Control
5. Entity Framework Core 8.0.6
6. NUnit.Framework as Unittest Project

## **Design patterns:**

1. Repository pattern
2. Unit of Work
3. Inversion of Control (IoC) and Dependency injection (DI)

## **Assumptions:**

* 1. **Total time spent**

|  |  |
| --- | --- |
| Setup environment and Architecture | 2 Hours |
| Implement Task + Unit Test | 2 Hours |
| Documentation | 1Hour |
| ci pipeline | 1Hour |