**Project & Tasks Management application**

**Solution structuring:**

The solution consists of 4 main folders: Core, Infrastructure, Presentation and Tests.

The Core folder contains two layers: Domain and Application.

The Infrastructure folder contains an Infrastructure layer.

The Presentation folder contains a Presentation layer, which includes two projects: BackgroundServices and WebApi.

The Tests folder contains a test project.

The Domain layer contains three folders: Constants, Entities, and Interfaces.

The Application layer contains four folders: DTOs, Interfaces, Mapper, and Services.

The Infrastructure layer contains four folders: DBContext, Migrations, Repositories, and UnitOfWork.

**Some SOLID principles applied:**

* Create a base entity class and make all entity classes derived from it.
* Create an ISoftDeletable interface that contain isDeleted prop and make all entities classes derived from it.
* Use the repository pattern to abstract data access.
* Use the UnitOfWork pattern.
* Create a base repository and make all repositories interfaces derived from it.
* Create an interface for each service and repository and unitOfWork.

**Some potential next steps to scale this into a microservices architecture:**

- A microservice can be built to manage users and roles.

- A microservice can be built to send notifications.

- A microservice can be built to send OTPs.

- An API gateway can be built to process requests and route each request to the appropriate microservice.

**Some security considerations applied:**

1. **Token handling:**

JWT (Bearer) is used for APIs security (Authorization and Authentication).

We have two primary Roles in the application (Admin and User).

1. **Input validation:**

Data Annotations is used for set validations on the all inputs in the application.

Ex: In ProjectRequestDTO class, Name property is Required