Main Points

1. Dependency Injection Using Constructor Injection (To Create Loosely Coupled Code)
2. Web API (Azure Authentication / Azure Key vault / Azure API Management)
3. Razor Page Used for Display UI Components and Data’s

Web API Data Details

1. Azure AD Authentication

I have added scope like read, write, put etc. for better security. (Create App Registration)

1. Created Dependency Injection to bind interface with services (Program.cs Class)

services.AddScoped<IInventoryService, InventoryService>();

(Used AddScoped because needs single instance of an Object through same Request)

1. Created One Abstract Layer (Repository Pattern)

Used Repository Pattern because it helps to reuse same code multiple times.

It makes simple when you wright Unit Testing

public interface IRepository<T>

    {

        IQueryable<T> FindAllDetails();

        void Create (T entity);

        void Update (T entity);

        void Delete (T entity);

}

public interface IInventoryRepository : IRepository<Inventory>

public interface IAccountRepository : IRepository<Account>

1. I have added validations. (Using data annotation and model state)

ModelState.AddModelError("inventoryID", "inventoryID model error").

return BadRequest(ModelState);

1. Used azure key vault to store sensitive information like Database connectionstring

ClientSecretCredential clientSecretCredential = new ClientSecretCredential(tenantId, clientId, clientSecret);

            SecretClient secretClient = new SecretClient(new Uri(keyvaultUrl), clientSecretCredential);

1. Used nlog for a better logging mechanism.

I have used ado.net code

Web Project

1. Use Razor Pages  (Convert Response Into Json File And Stringify It On Html Page)
2. Get a response from web api and display result in UI(Razor Page )