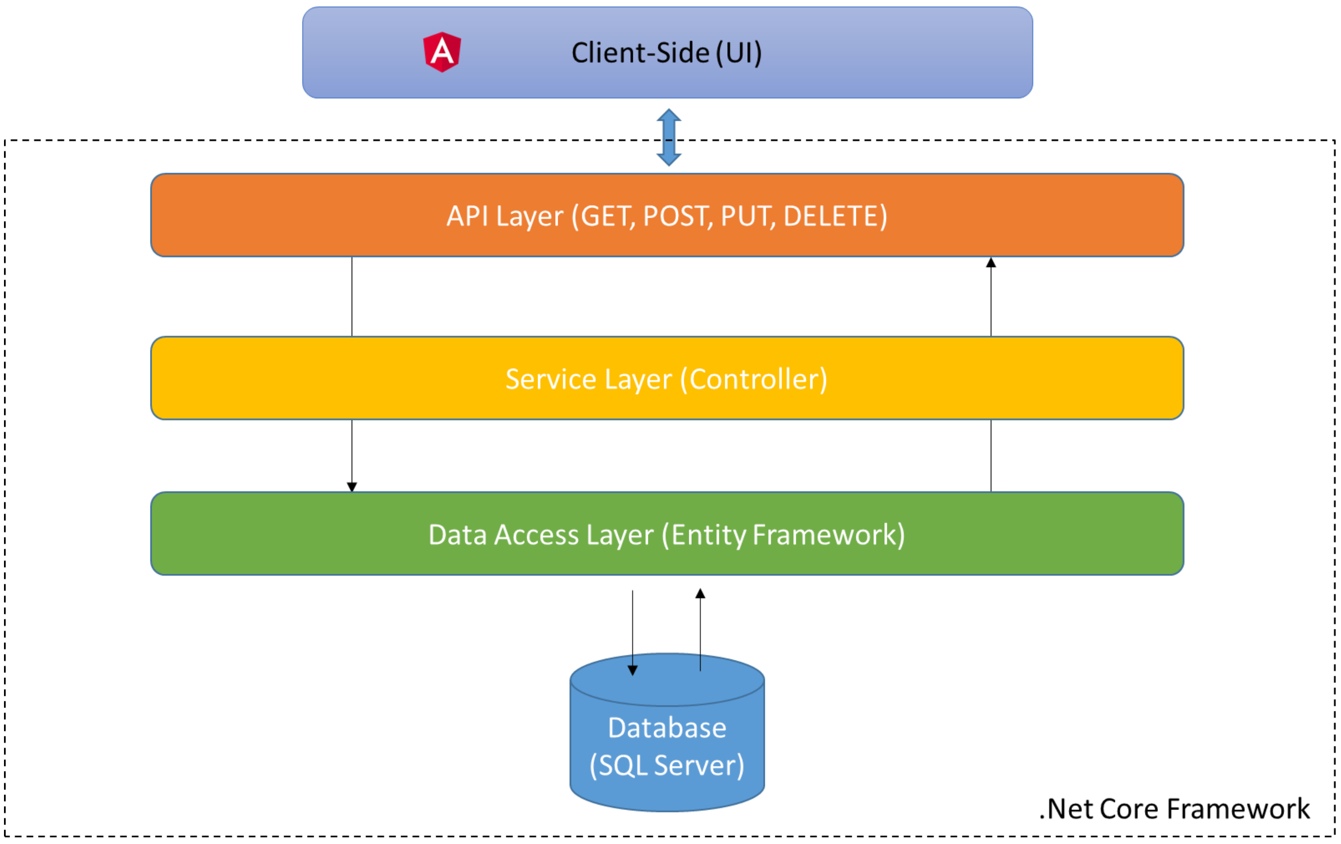
The Solution Design

The whole system adopts three layers to design, seeing Figure 1. It includes the API Layer, Service Layer and Data Access Layer. As you know, NET Core is a general-purpose development framework that can be used to build modern, scalable, and high-performance cross-platform software applications, so this framework is chosen to develop the whole back-end application, the database uses Microsoft SQL Server to provide the data storage. In terms of the client-side, the front-end framework Angular is used to build single-page client applications.



Client-Side (UI)
API Layer (GET, POST, PUT, DELETE)
Service Layer (Controller)
Data Access Layer (Entity Framework)
Database (SQL Server)
.Net Core Framework

Figure 1. The system structure of 'DutchTreat'

The API Layer is responsible for providing interfaces to data or functionality of one or more applications for the client, mainly including user login or logout, gaining product list and generating orders. Considering protected resources, the authentication and authorization scheme adopts JwtBearer authentication.

The Service Layer provides the implementation of access data and related business logic. It includes authenticating user login, retrieving product information from the database, and saving the order to the database.

The Data Access Layer focuses on communication with the database. There are four types of entities, including Order, OrderItem, Product and StoreUser. Entity Framework Core is used as an object-database mapper. It supports LINQ queries, change tracking, updates, and schema migrations.

Identify the Problems and How to fix it.

1. The entity of StoreUser does not contain a definition for ‘Email’ and a definition for ‘UserName’.
2. The type 'DutchTreat.Data.Entities.StoreUser' cannot be used as type parameter 'TUser' in the generic type or method 'IdentityDbContext<TUser>'. There is no implicit reference conversion from 'DutchTreat.Data.Entities.StoreUser' to 'Microsoft.AspNetCore.Identity.IdentityUser'.

Fixed 1 & 2: The entity of StoreUser should be inherited from 'Microsoft.AspNetCore.Identity.IdentityUser'.

3. 'NullMailService' does not implement interface member 'IMailService.SendMessageWithAttachment(string, string, string, string[])'.

Fixed 3: Add the implementation of the interface member ‘IMailService.SendMessageWithAttachment’ of the NullMailService Class.

4. The Database of ‘DutchTreatDb’ is not created.

Fixed: Using migrations to create this database, then run the executable file “DutchTreat.exe” with the argument “/seed”.

5. When using the test account to log in to the system, Connect the Database of ‘DutchTreatDb’ failed.

Fixed: Switch to the development environment by modifying the value ASPNETCORE_ENVIRONMENT in the environmentVariables node in the profiles node in launchSettings.json.

6. ‘cart’ is not a known element in the file “shopPage.component.html”

Fixed: Add the declaration of the component of ‘CartView’ in the file “app.module.ts”.

7. The home page cannot be presented.

Fixed: Modify the name of the selector of the AppComponent that is “app-root”.

8. When using the test account to log in to the system, return “Failed to login”.

Fixed: The problem was caused by the “return” keyword losing in the post action of login in the file “AccountController.cs”

9. Unable to resolve service for type 'AutoMapper.IMapper' while attempting to activate 'DutchTreat.Controllers.OrdersController'.

Fixed: Using AddAutoMapper() method in ConfigureService() method of the startup.cs class.

10. Cannot insert explicit value for identity column in table 'OrderItem' when IDENTITY_INSERT is set to OFF.

Fixed: Before adding existing entities to “DbContext”, call Attach method to these entities.

Unit Testing Controllers

Method	**Test case name**
ProductsController.GetAllProducts	GetAllProducts_ShouldReturnOkResponse_WhenDataFound
	GetAllProducts_ShouldReturnOkResponse_WhenDataNotFound
	GetAllProducts_ShouldReturnBadRequest_WhenThrowException
OrdersController.GetOrderById	GetOrderById_ShouldReturnOkResponse_WhenValidInput
	GetOrderById_ShouldReturnNotFound_WhenNoDataFound
	GetOrderById_ShouldReturnBadRequest_WhenThrowException
OrdersController.GetOrderByUser	GetOrdersByUser_ShouldReturnOkResponse_WhenValidInput
	GetOrdersByUser_ShouldReturnBadRequest_WhenThrowException
OrdersController.CreateOrder	CreateOrder_ShouldReturnCreatedResponse_WhenValidRequest
	CreateOrder_ShouldReturnCreatedResponse_WhenInValidRequest
AccountController.CreateToken	CreateToken_ShouldReturnCreatedResponse_WhenValidRequest
	CreateToken_ShouldReturnCreatedResponse_WhenInValidRequest