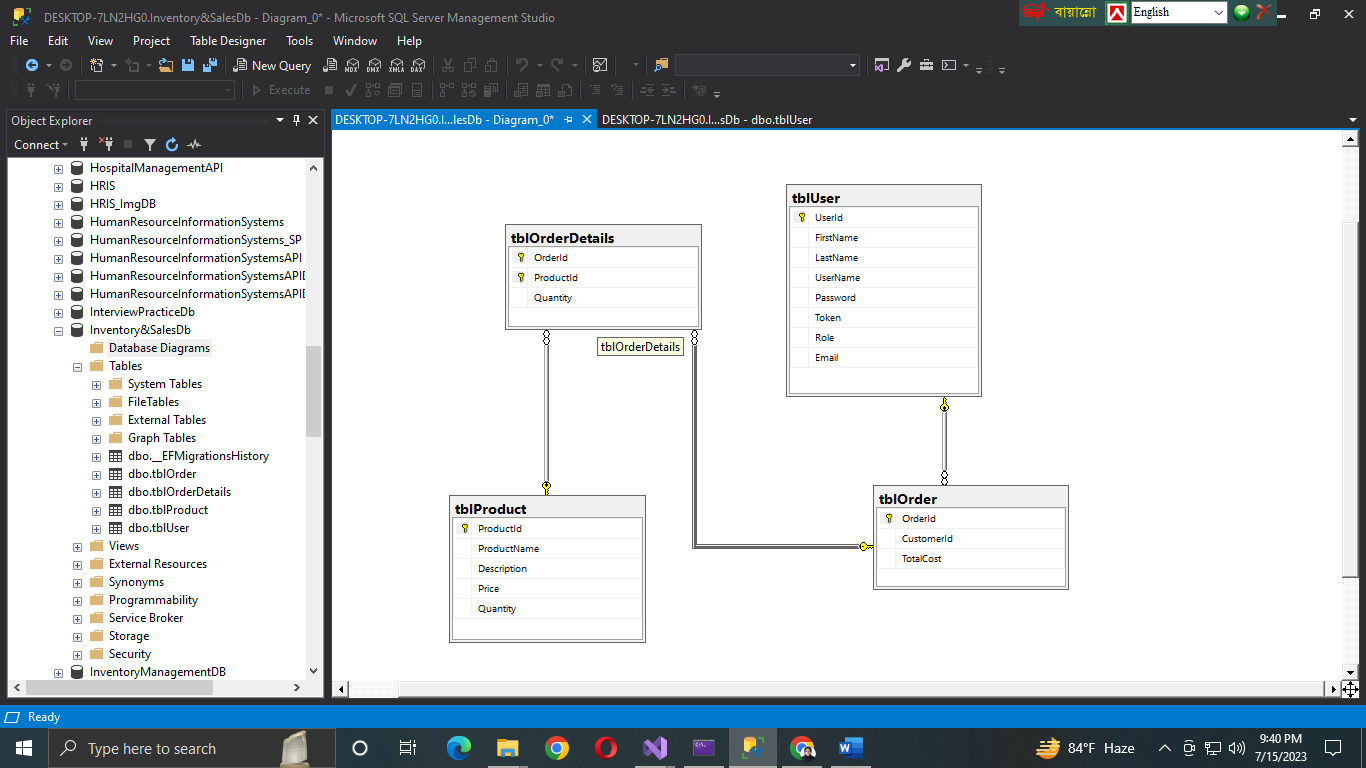
**PROJECT TITLE**

**INVENTORY & SALES MANAGEMENT SYSTEM**

DATABASE DESIGN

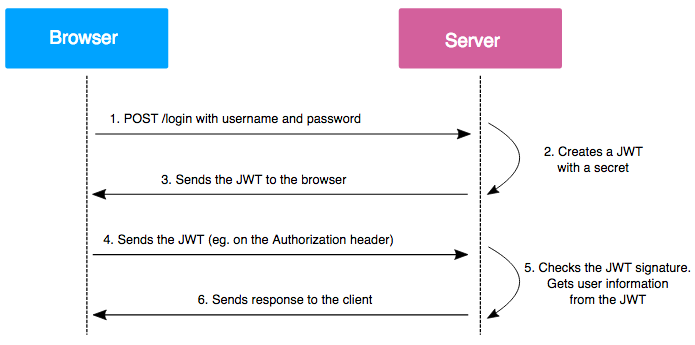
**Tables in the database**

1. tblProduct
2. ProductId
3. ProductName
4. Price
5. Description
6. Quantity
7. tblOrder
8. OrderId
9. CustomerId
10. TotalCost
11. tblOrderDetails
12. OrderId
13. ProductId
14. Quantity
15. tblUser
16. UserId
17. FirstName
18. LastName
19. Email
20. Role
21. Token

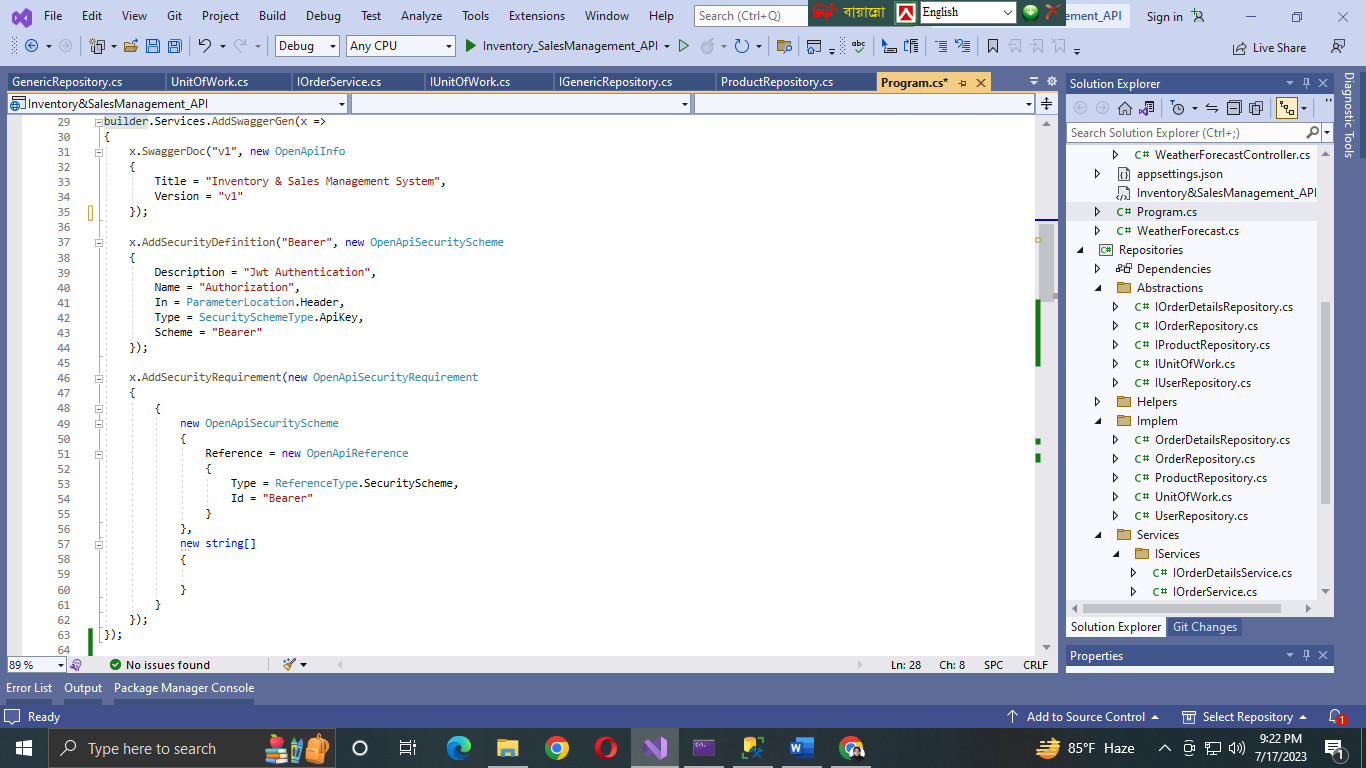
**Notes :**

1. tblProduct is for Inventory Management.
2. tblOrder and tblOrderDetails are for Sales Management.
3. tblUser is for Authentication and Authorization.
4. The users are playing the role of customers.

**AUTHENTICATION & AUTHORIZATION**

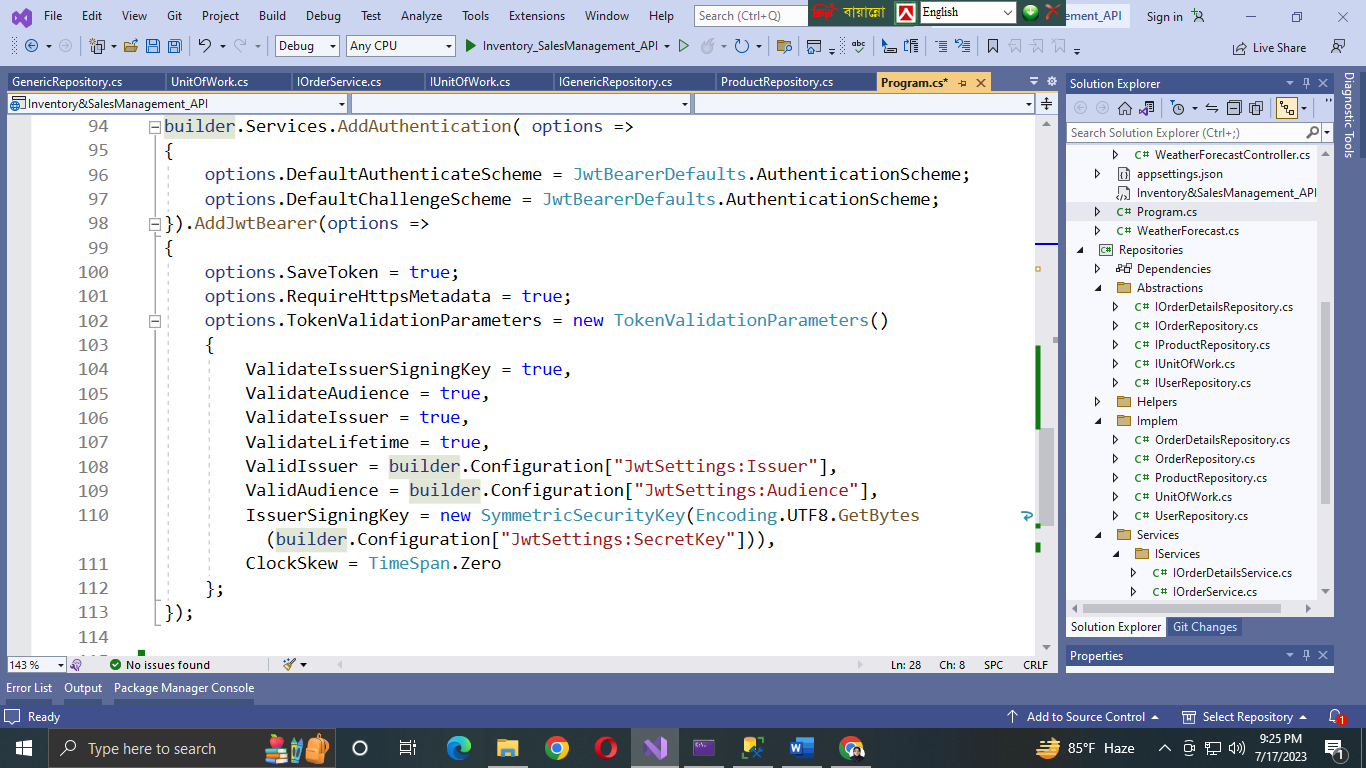


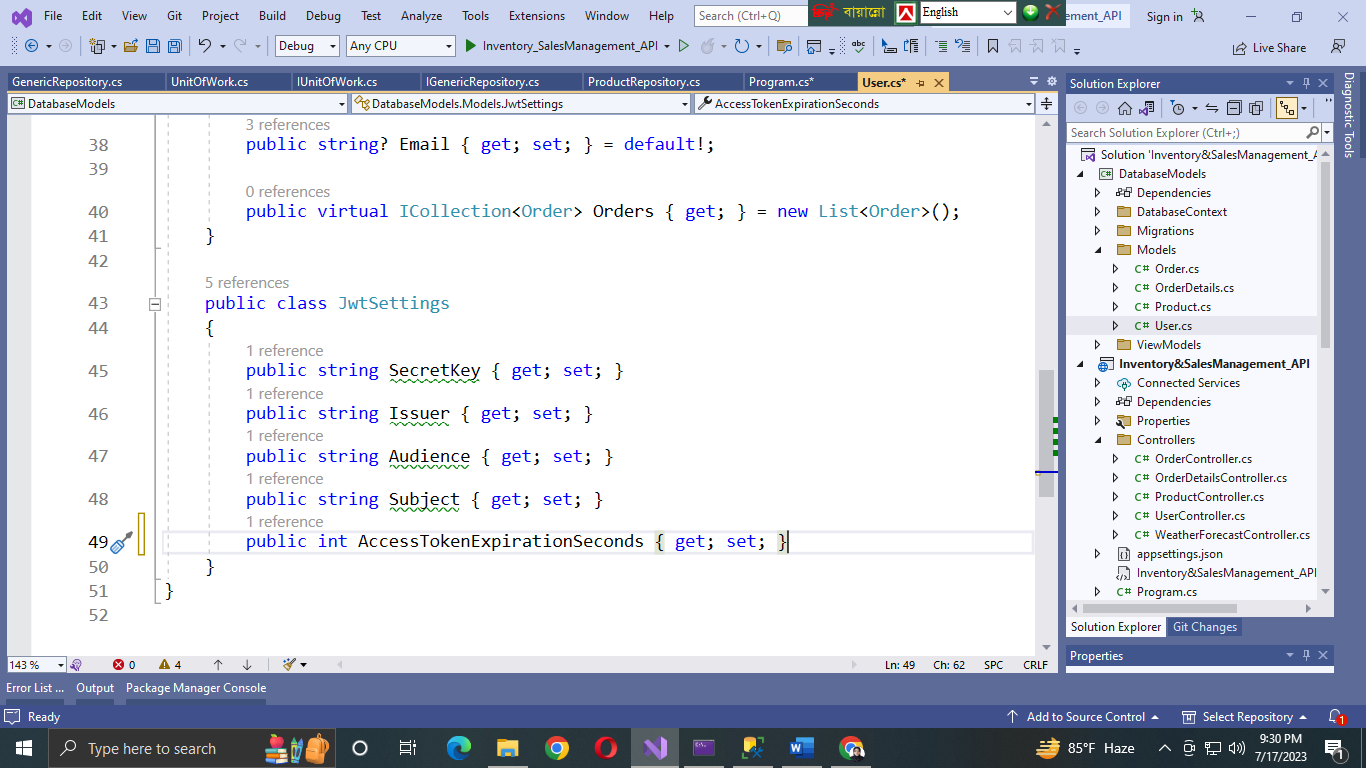
Three main methods for JWT token SWAGGER integration

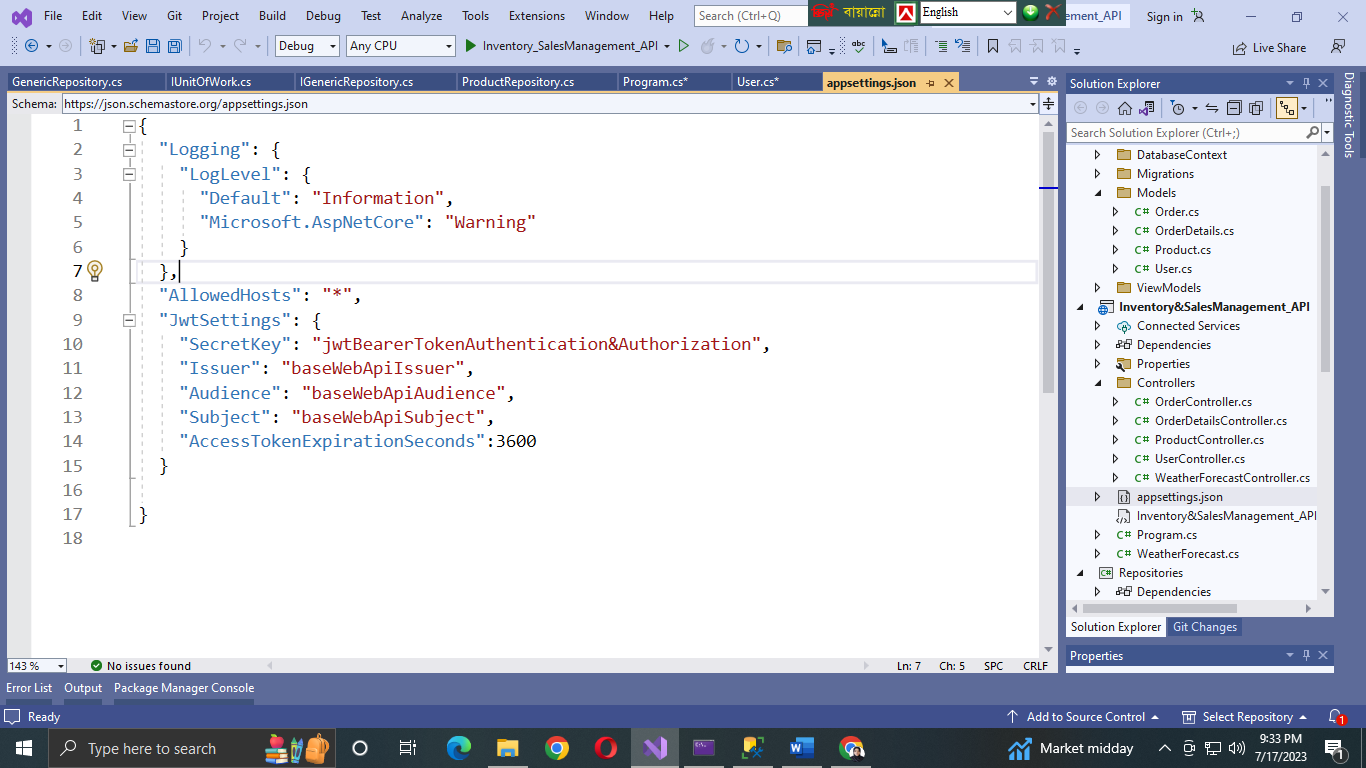
1. AddSecurityDefinition()
2. AddSecurityRequirement()

Package : Microsoft.AspNetCore.Authentication.JwtBearer

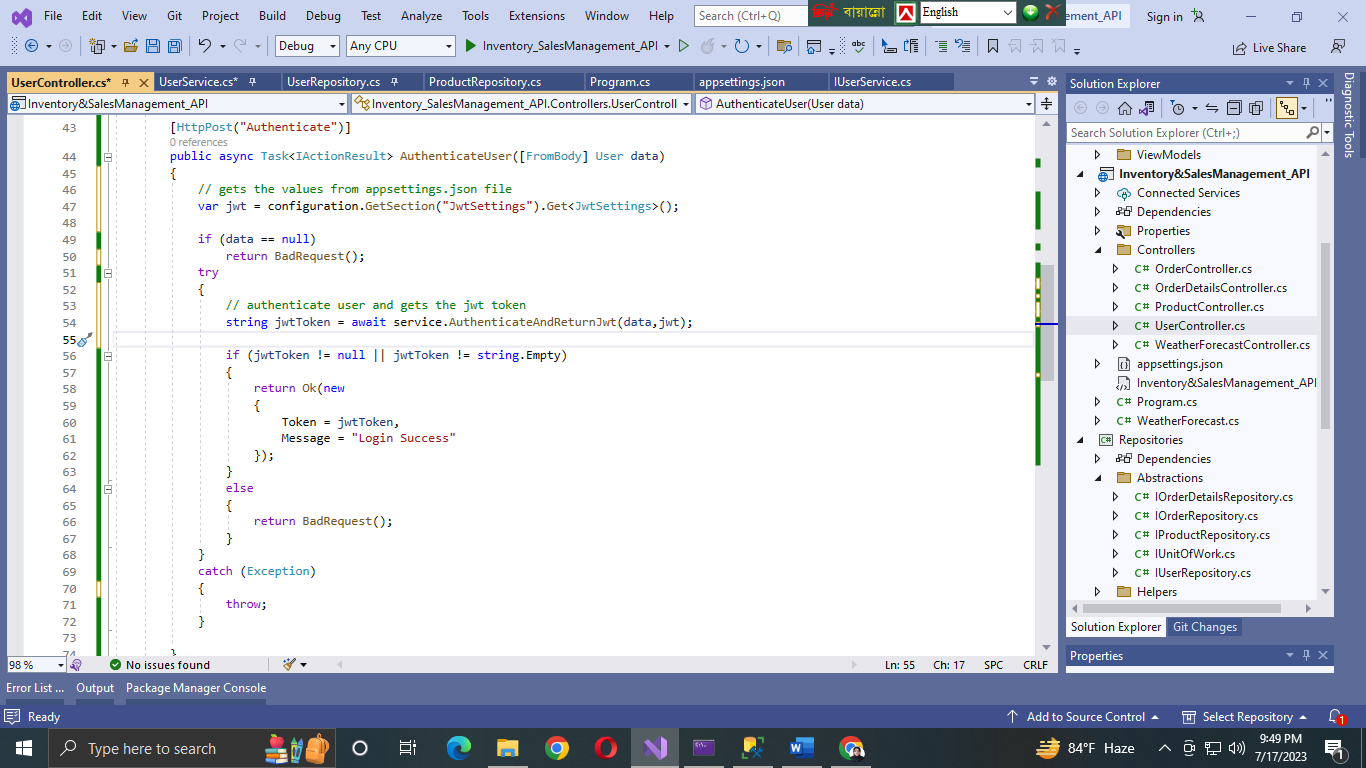
If the username and password match with the stored user data in the database, then a token will be generated and user will be able to access resources through that token.

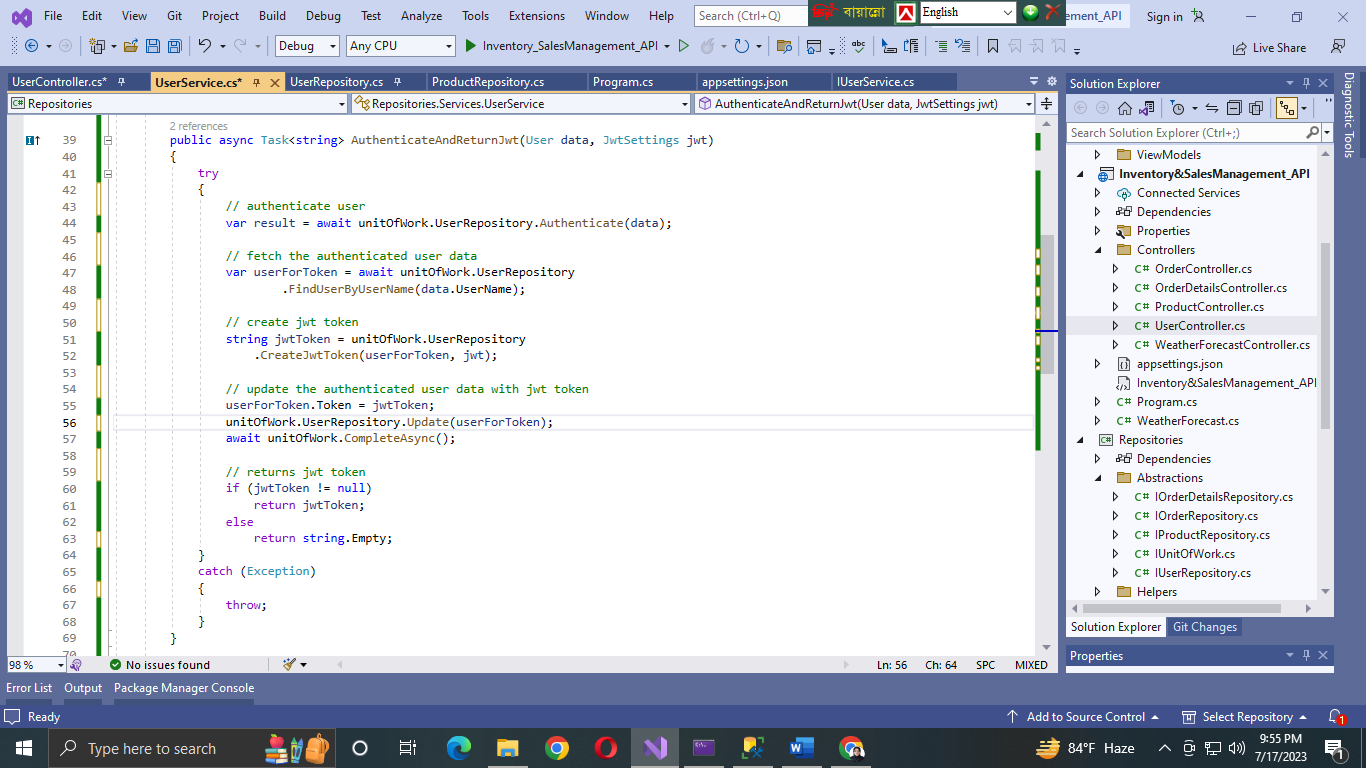
Method for JWT Bearer authentication : Services.AddAuthentication()

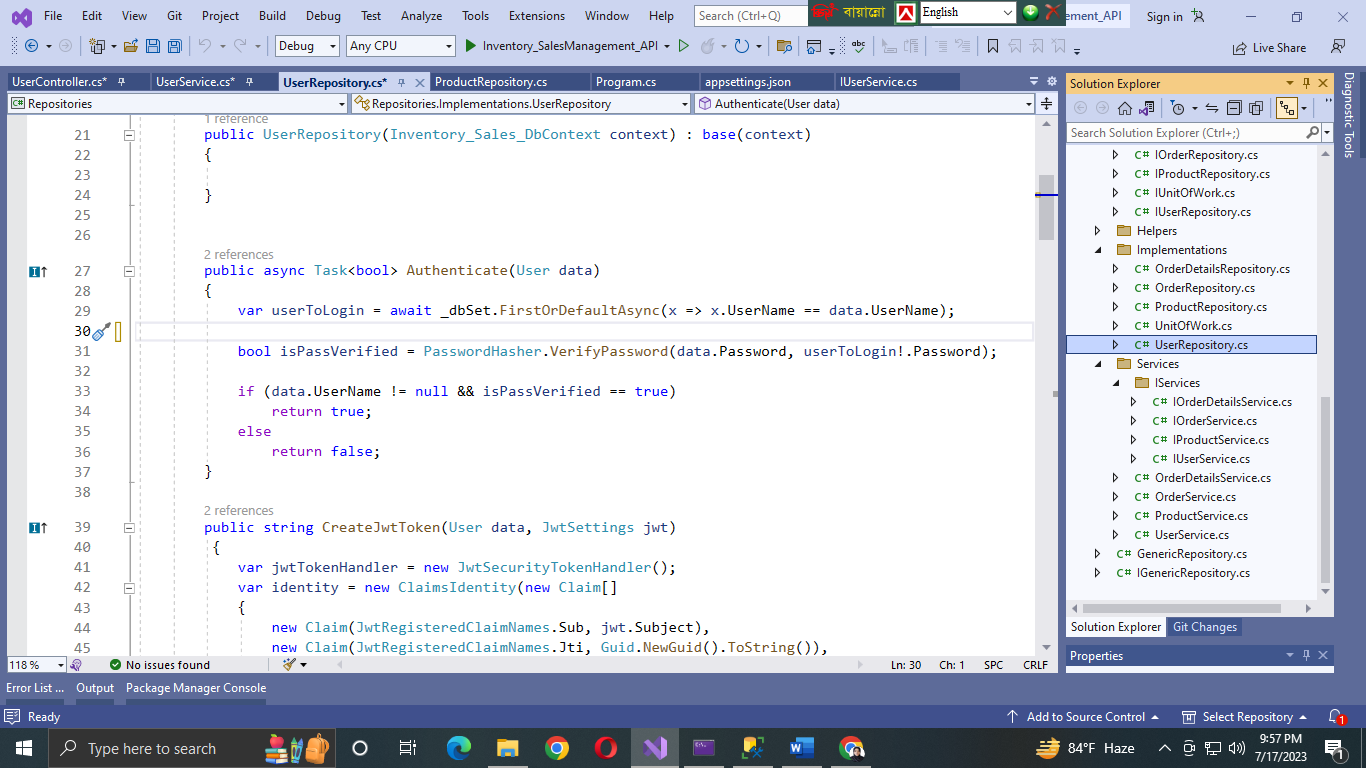
To pass the values in program.cs file I have created a class named “JwtSettings”

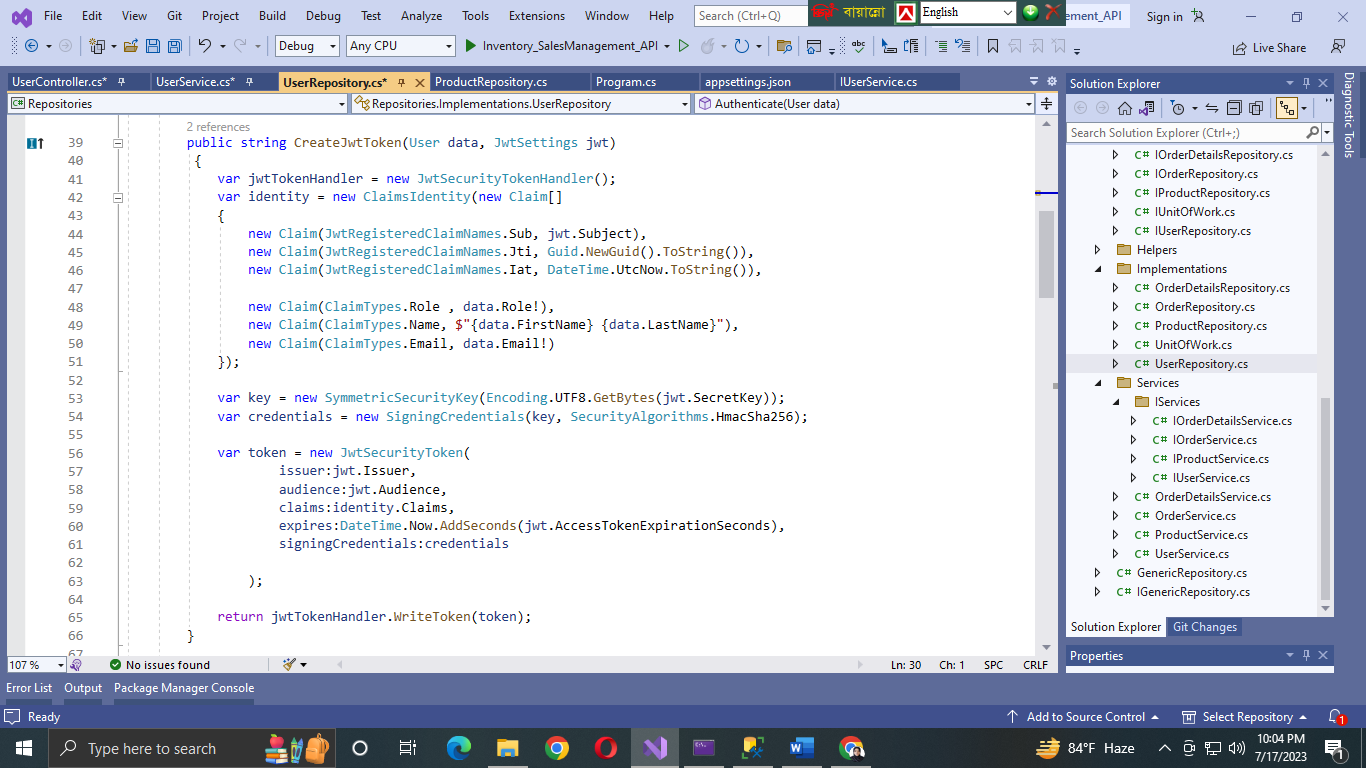
The values of the properties of “JwtSettings” class are provided through appsettings.json file.

When a user provides the username and password to log in, if those credentials are correct then the server generates a JWT token for that client/user.

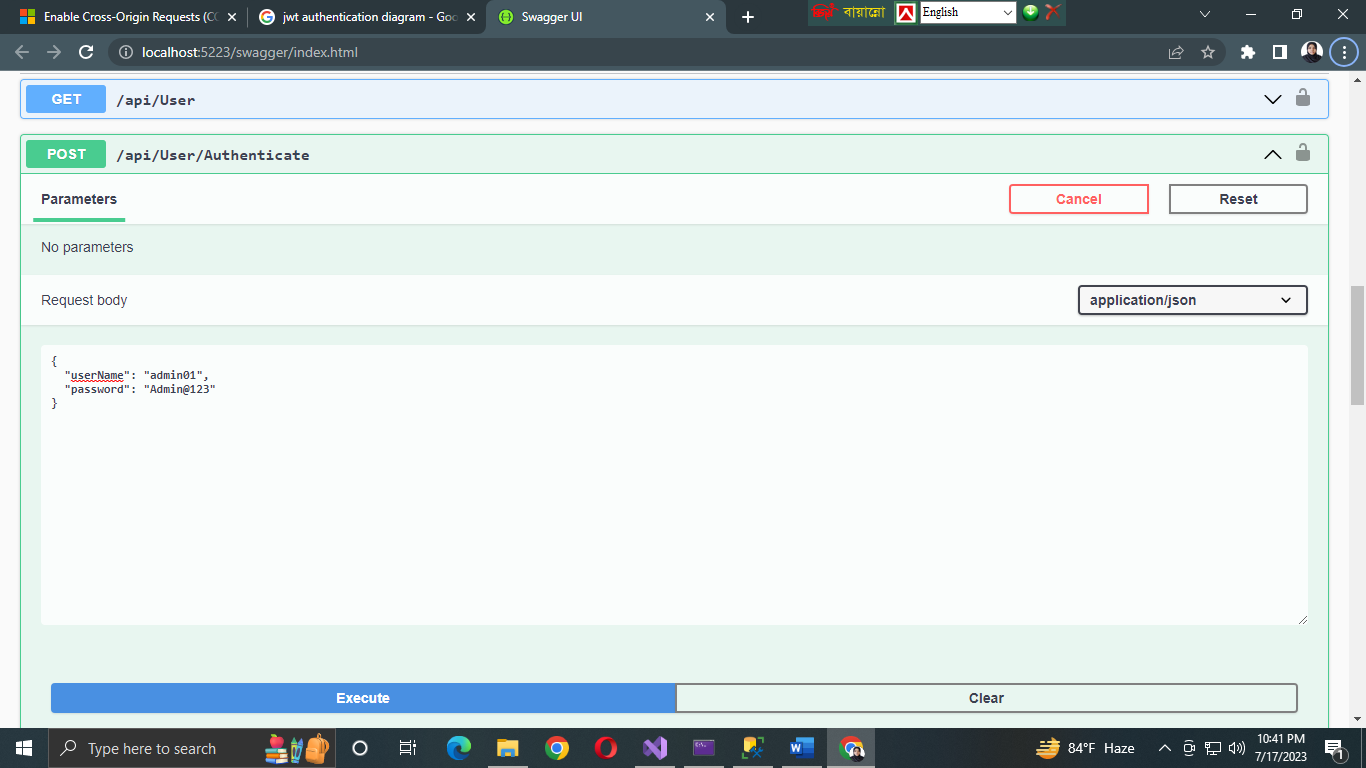
**User Controller :**

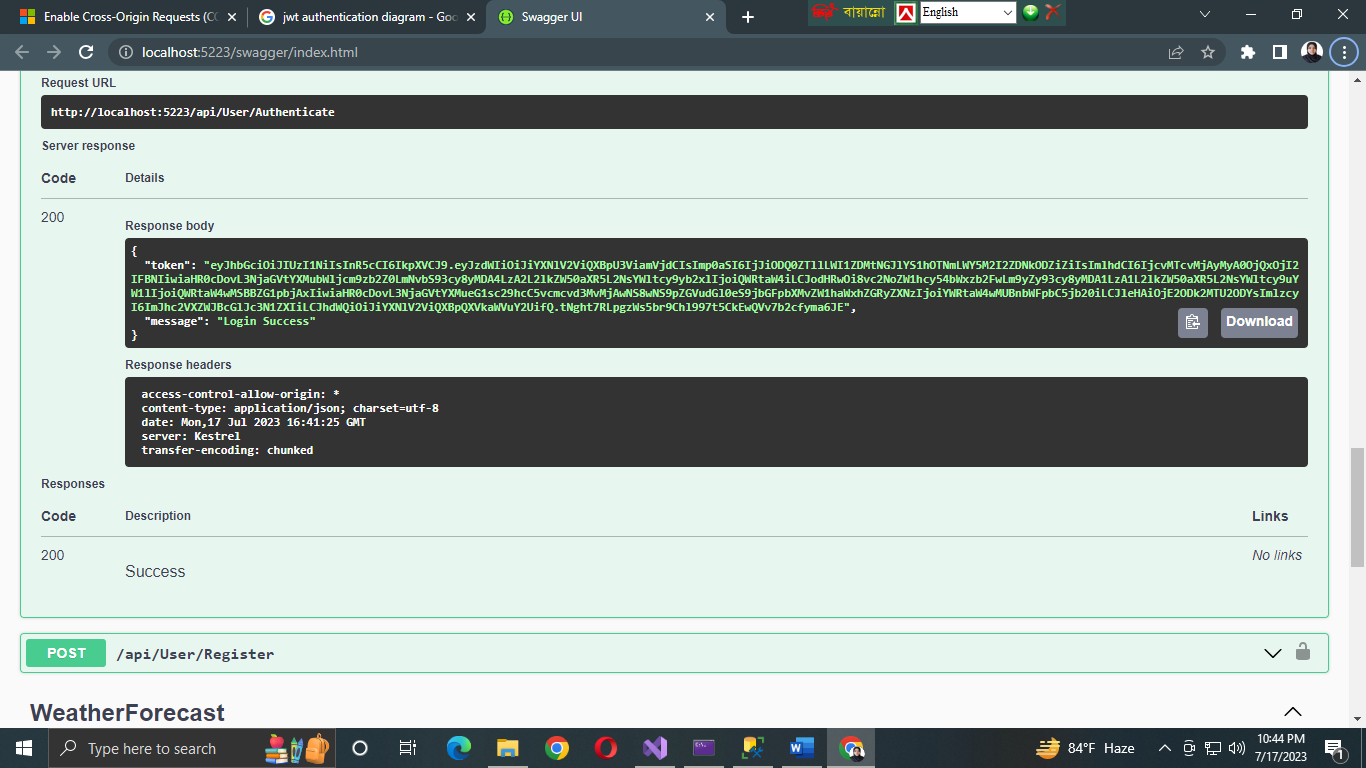
**User Service :**

**User Repository :**

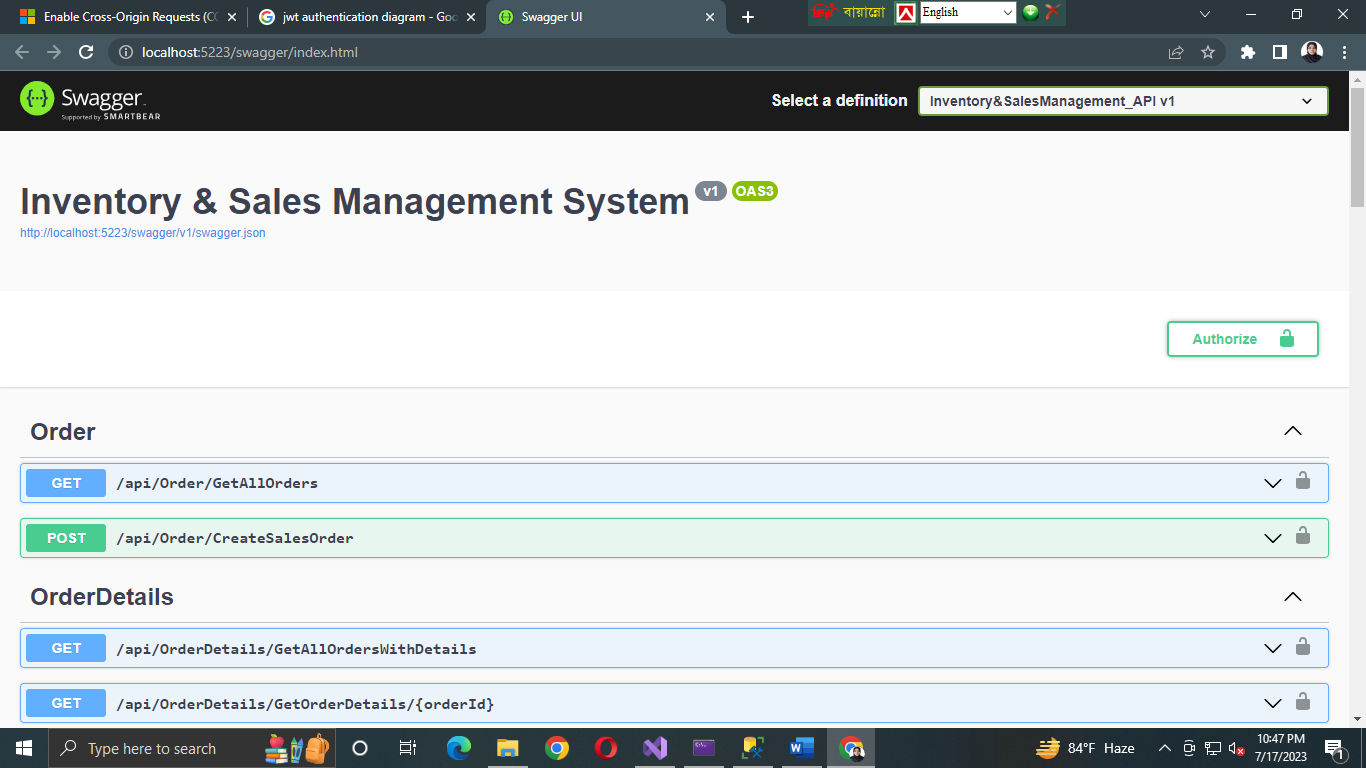
The JWT token is also created in User Repository.

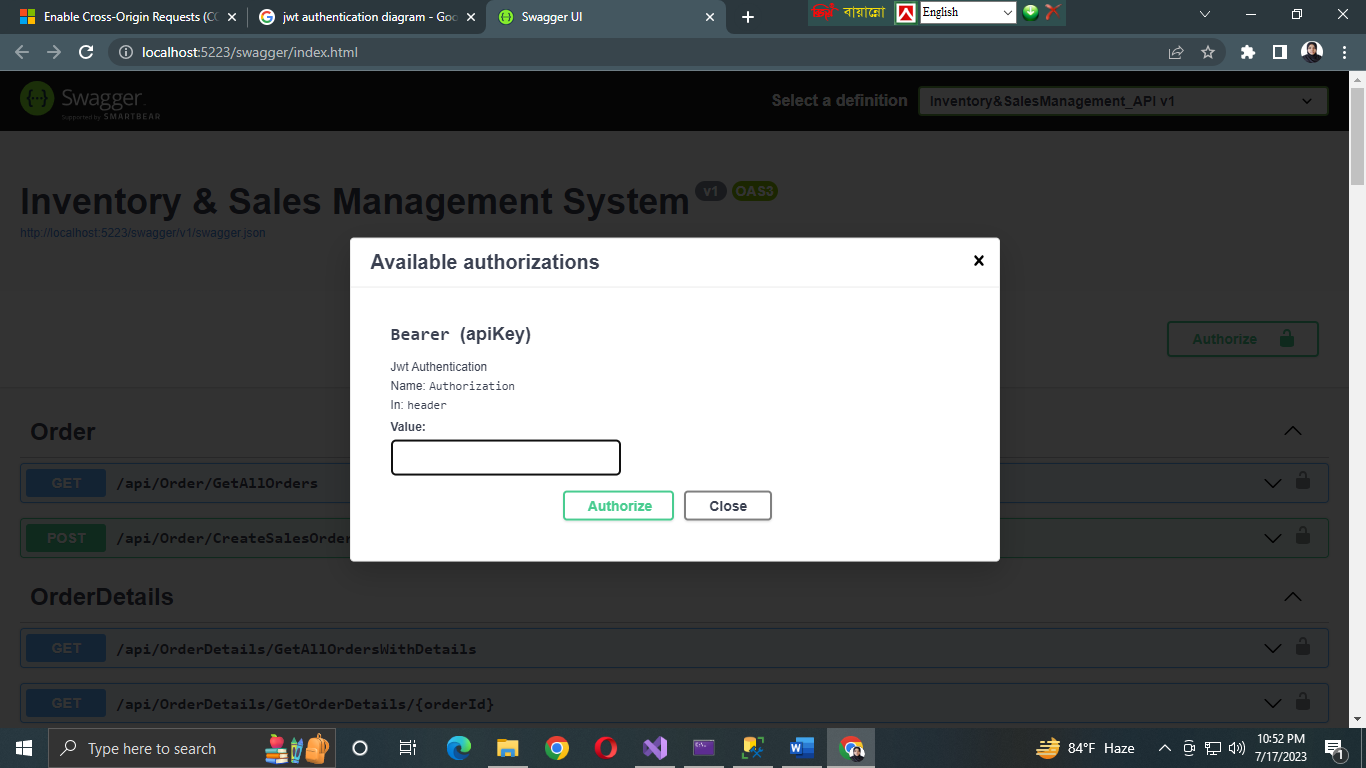
Login and Token integration :

1. User will provide username and password to log in.
2. Server will verify the credentials and if verified then will authenticate the user and will generate a token.



1. User will append the token in the authorization header.





This token will remain valid till the expiration time. The token expiration time is provided in appsettings.json file.

I have also applied RBAC in Product Controller only.

**REPOSITORY PATTERN**

**Controllers**

**Unit of work**

**Repositories**

**Data Source (Database)**

**Services**

**Notes:**

* Through the Unit of work, we can access any repository as necessary.
* The methods in the Generic Repository are common for all and can be accessed by any repository.
* But the entity-wise repositories are unique for that entity. For example, if we need to create any method that is necessary only for the Product, we will create that method in Product Repository, not in Generic Repository. All the methods of Product Repository and Generic Repository will be accessed from Product Service through the instance of Unit of Work.