Authentication Blueprint:

1.Design Database with Identity user with Aspnetcore.identity, This has inbulit fields for user creation with Roles inbuilt, If want to add new field you can do by generating a new class which inherits from Identity User.

2. Registration Blueprint:

* Create a customized registration model, hit the registration API, Prepopulate Application User and save the data with userManager.CreateAsync(user,registrationRequestDto.Password).
* If there is no role assigned from frontend assign to the default role. Then hit the API services with the email and role, For our module email and Username is same and unique, check for the user, If role exist then \_userManager.AddToRoleAsync(user, roleName), Or else first create a role by \_roleManager.CreateAsync(new IdentityRole(roleName)).GetAwaiter().GetResult() then execute the previous step.

3. Login Blueprint

* Hit the API controller with Login model
* Request the User with Userid
* Check isvalid with userid and password \_userManager.CheckPasswordAsync(user,loginRequestDto.Password)
* Get Role with user
* Generate token with tokengenerator, assigning id, email and username in JwtRegisteredClaimNames and roles in claimtypes, then create token and return that token back populating that in model
* Set the token in the context

4. Design Base Service to access from front end to each API and authorize.

* Design a request Dto (Content/Data in form of object, Contenttype, Url, Access Token, ApiType)and request a response DTO(IsSuccess, Messesge, Content in form of Object)
* Set the API Link in the Appsettings, Populate the values in the form of an object in program.cs.
* Populate ResponseDto and call SendAsync(RequestDto requestDto, bool withBearer = true)
* Assign the bearer token in header, Assign Url, Assign content, Specify content type, Deserialize content and send Data, For any kind of Get request populate data in Url.
* Deserialize the request.

5. Registering a coupon

* Design a coupon API which contains interacts with coupondatabase (Coupon Code, Min Amount and discount amount)
* Only admin have access to create, edit and delete coupon
* Design a MVC in Web Project to access the API project through base service.

6. Design a Product Database

* Design a productAPI which has a product database attached and we design Post, Put , delete and Get API.
* Design a front and to do all these things through base service.

7. Home page and login

* FOLLOW Login Blueprint
* Retrieve the email, Type, Name, Role in identity from the jwt token.
* Check the User role User.IsInRole(SD.RoleAdmin) and populate the options, Check

!User.Identity.IsAuthenticated then Populate option for LOGIN/ Registerv or populate Username / LogOut.

* During the login Populate the full Product details one by one
* Then click on anything Details, Call the Product API through base service and get the product details.

8. Add to Cart method

* Form a CartHeader model, With HeaderId, UserId, Coupon code, Total amount and discounted amount, form a Cartdetail model with CartdetailId, Cart header as a Foreighn key, Product , Count.
* When Addtocart method is hit retrieve the UserId from jwt token populate it in cartheader and productid and count in cartdetails, Wrap this in a combined model which can consist a list of Cartdetails with different count.
* . Design an CartAPI with CartHeader model, With HeaderId, UserId, Coupon code, Total amount and discounted amount, form a Cartdetail model with CartdetailId, Cart header as a Foreighn key, Product , Count, where there will be tables for cartheader and cartdetails.
* Check for if any cart is present for the user, If not then first Map and insert that record.

Then assign that cartheaderid in cartdetails, map it and insert it in DB,

If already cart present for that user if any count of that product is already inserted, if not then Add the cartdetails, Or else update it.

* Design GetCart with UserId method .

9. Cart checkout and place order

* Login with a Particular User, Click on Cart and then Click on “Looks Good”, CartIndex open, Fill Email. Name and Phone( Address also required for delivery), Then Click on Place order.
* Order API needs to be created, OrderDetail and OrderHeader (consist of List of Order details)are the two model similar as Cartdetails and cartheader, Thus we hit the checkout with CARTDTO for a particular user, Create order by mapping those to OrderDetail and Orderheader, adding the extra parameter properly in the mapperconfig. Save those order detail and get the OrderHeader model back with List Orderdetails product specific.
* Install Stripe.Net, Create a Stripe account and get an API Secret Key, Save that in Appsettings.json, now create a StriperequestDTO with SuccessUrl(where API hit after Pay success with the OrderheaderId ), CancelUrl(to hit the checkoutIndex method).
* Now Hit the CreateSession with StriperequestDTO, create session with Approval Url, CancelUrl, LineItemoptions with Price and count of each with product name(Check code), also discount is added in the session, then sessionUrl is created and saved in the dB, Session Url populated in the striperequestDto and returned back, Link redirected to session Url.
* After confirmation, Payment Id odtained from session (refer code). Saved in Orderheader table and marked approved.