**Shopping Cart Demo (SCD)**

I have created a shopping portal in microservice architecture and tried to cover all basic areas in Azure Cloud. Below is the workflow of my demo application:

A diagram of a shopping cart

AI-generated content may be incorrect.

**This solution is microservice architecture**

## Angular UI (SCD\_UI)

* Developed using Angular:20.x
* Will be expose on <http://localhost:4200>
* Key pages:
  + - Sign In
    - Signup
    - Product Dashboard (MyShop)
    - View Cart
    - Checkout
    - My Orders

**Note**: just for quick login customer details are prefilled in login screen for Customer ([customer@nagarro.com](mailto:customer@nagarro.com) / Pass@123) and for Admin ([admin@nagarro.com](mailto:admin@nagarro.com) / Pass@123) Further you can sign up as Customer/Admin role with correct email address to get emails and the same credentials will work immediately for sign in.

## Ocelot API Gateway (SCD.GatewayProject)

* Gateway base url <http://localhost:7000>
* Upstream url will be <http://localhost:4200>
* Downstream url be
  + - For Product: <http://localhost:7003>
    - For Cart: <http://localhost:7004>
    - For Orders: [http://localhost:7005](http://localhost:7004)

## .Net Core Api (SCD.Services.AuthAPI)

* Developed using .Net Core Web App (net 8.0)
* Will be expose on <http://localhost:4200>
* Key Action Methods:
  + - Register
    - Login

## .Net Core Api (SCD.Services.ProductAPI)

* Developed using .Net Core Web App (net 8.0)
* Will be expose on <http://localhost:7003>
* Key Action Methods:
  + - Get (get all products)
    - Get/{id} (get product by id)
    - Post(Add products) for ADMIN role only

## .Net Core Api (SCD.Services.CartAPI)

* Developed using .Net Core Web App (net 8.0)
* Will be expose on <http://localhost:7004>
* Key Action Methods:
  + - CartUpsert (Add/Update Items in Cart)
    - RemoveCart (Delete items from cart)
    - GetCart/{UserId} (Get cart by userid)
    - ClearCart (Clear cart)

## .Net Core Api (SCD.Services.OrderAPI)

* Developed using .Net Core Web App (net 8.0)
* Will be expose on <http://localhost:7005>
* Key Action Methods:
  + - CreateOrder (Place a order for cart items)
    - GetOrders/{Userid} (Get all orders list for logged in user)
    - GetPendingOrders(List all newly created orders) for ADMIN role only
    - UpdateOrderStatus(Admin can approve/reject order) for ADMIN role only

## .Net Library (SCD.MessageBus)

* Utility library to send message in Azure bus Queue
* This library is reference from CartApi and OrderApi send message in queue

## Azure Function App (SCD.EmailProcessorFunction)

* This is ServiceBusTrigger azure function to read messages from azure message bus queue
* Once we receive any messages, we create a mail template (to, subject, body) and send message to Azure Logic App to send mail
* Also send a new message to read BlobStorageQueue if admin as approved this order, so that next order can be set as Delivered status.

**Features**

**Customer Features and Workflow:**

* User Sign up as Customer
* User Sign in as (Customer) and get Auth token (JWT)
* MyShop (List of products) page/Add product to cart
* Checkout Cart
  + Inter service call to CartAPI to get Cart Details
  + Inter service call to ProdctAPI to get Product Details
  + Insert Create Order
  + Inter service call to CartApi to clear cart
  + Send Order confirmation Mail to customer from Azure Logic App
  + Redirect to Order confirmation page which will show order number and link to Orders list page
* User can track order from order list

**Admin Feature and workflow:**

* User Sign up as Admin
* User Sign in as (Admin) and get Auth token (JWT)
* MyShop (List of products) page to view all products
* Add Product (To register new products)
* Pending Orders
  + This page will list all new order places
  + Admin can Approve or Reject the order
  + Send Message in azure message bus queue

**Azure Function:**

* Read Azure Message Bus queue
* Send Approved/Reject Mail to customer from Azure Logic App
* If order is Approve send a message in Blob Storage Queue

**Azure Storage Queue Worker (in OrderAPI):**

* Read message
* Update order status to (Delivered) assuming that process between order status and order delivery is done.
* Send Delivered Mail to customer from Azure Logic App