Low Level Design for EShoppingCart

Folder Structure (3 Layer Architecture)

Folder	Role
Controllers	Accept API requests, call services. (Presentation Layer)
Services	Contains logic like placing orders, managing wallets, etc. (Business Logic Layer)
Repositories	Direct database operations like fetching products, users, orders. (Data Access Layer)
Models	C# classes representing database tables.
Context	Contains AppDbContext which manages EF Core’s database connection.
Migrations	Stores EF Core database migration files.

Tech Stack Used

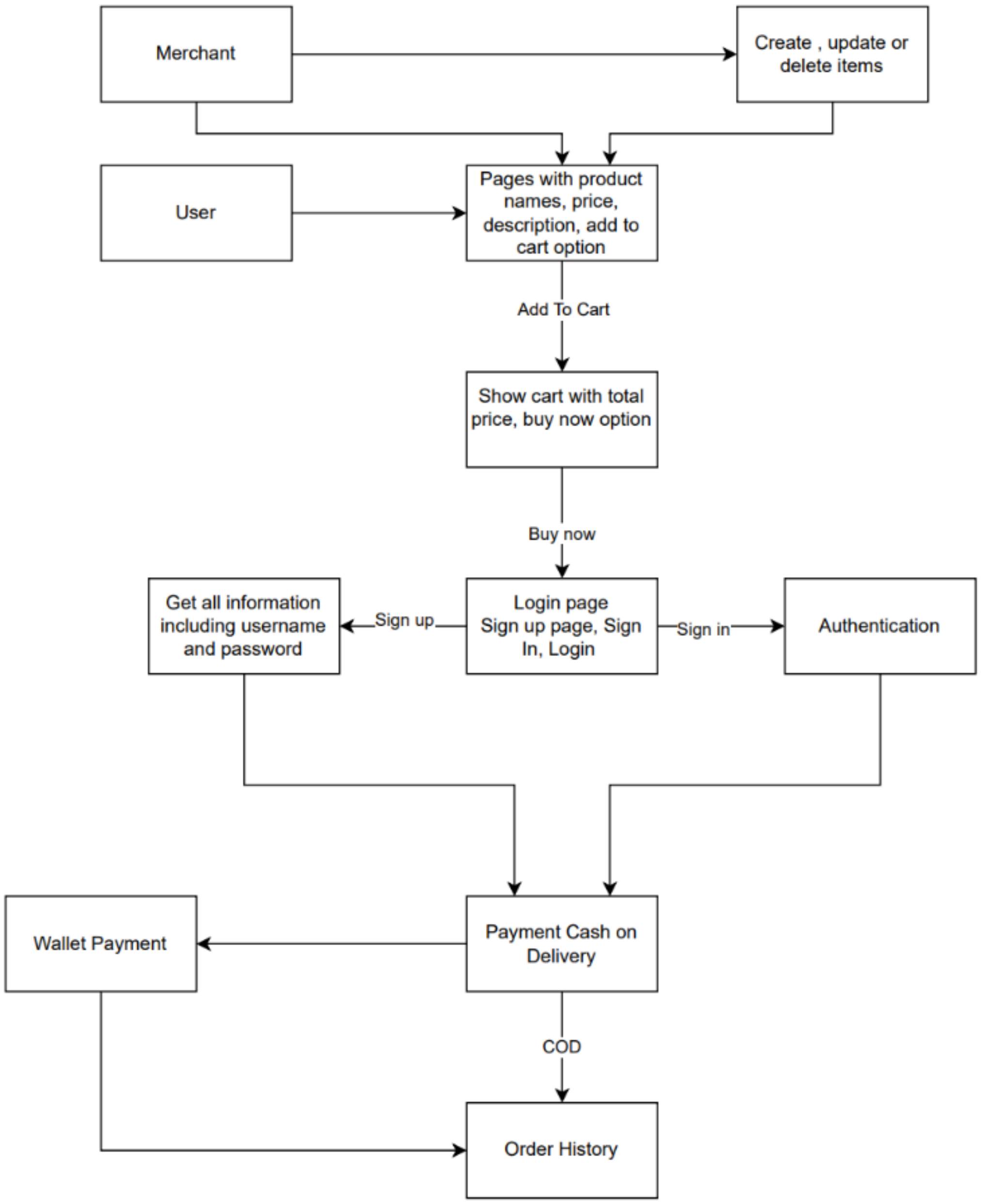
Layer	Technology	Purpose
Presentation Layer (API Layer)	**ASP.NET Core Web API, Swagger**	Handles incoming HTTP requests from users, exposes endpoints for operations like viewing products, adding to cart, login, payment, etc. Swagger provides UI to test these APIs
Business Logic Layer (Service Layer)	**C# Classes in Services and Business folders**	Contains the main business rules — how orders are placed, wallets are updated, validation of users, etc. It connects controllers with repositories.

Layer	Technology	Purpose
Data Access Layer (Repository Layer)	**Entity Framework Core, LINQ, SQL Server (SSMS)**	Interacts directly with the database. It performs CRUD (Create, Read, Update, Delete) using repositories and AppDbContext.
Database	**Microsoft SQL Server**	Stores all data — users, products, orders, carts, wallets, and admin details.

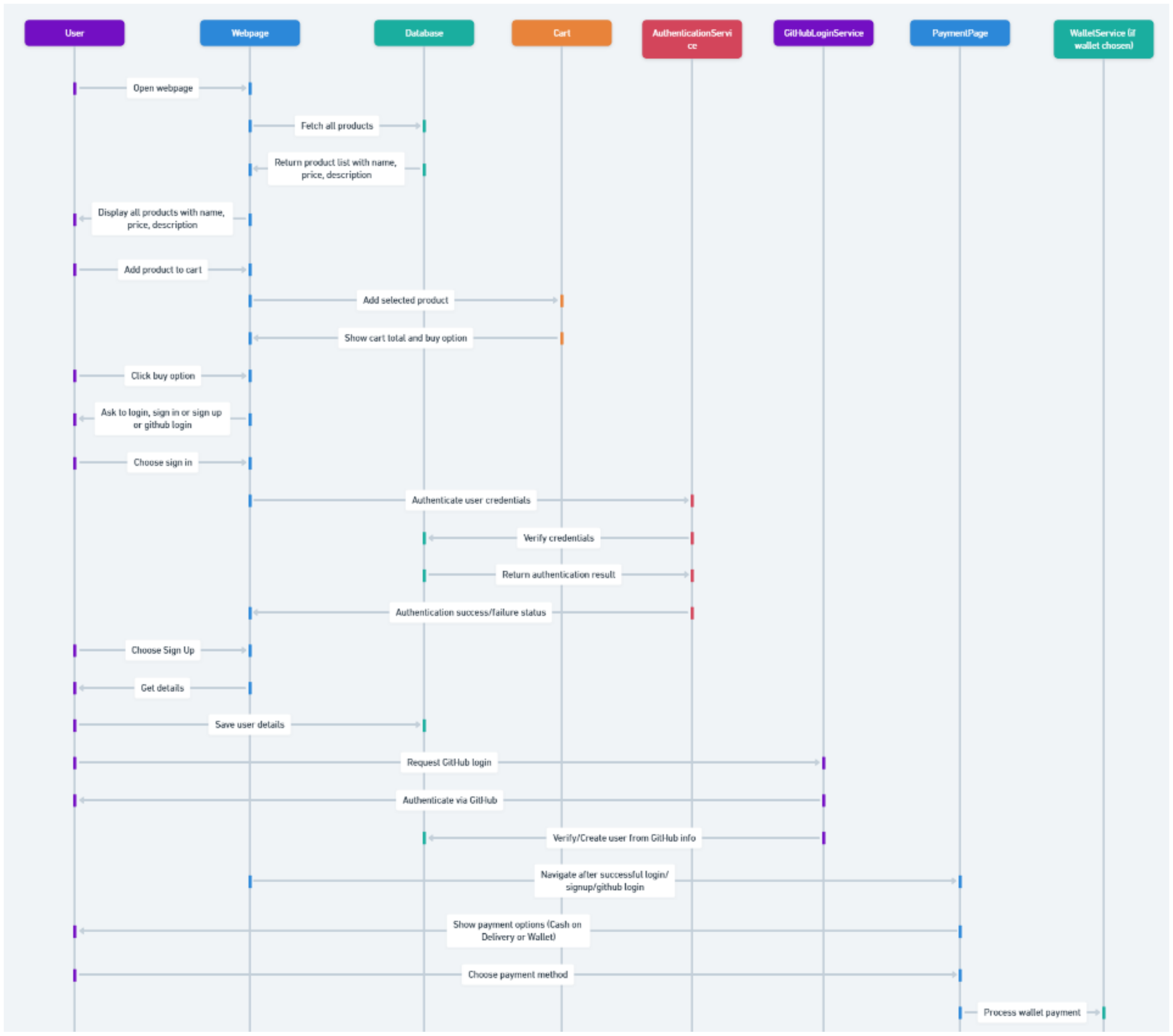
Tools Used

Tool	Purpose
Swagger UI	For testing API endpoints (since there's no front-end).
Entity Framework Core (EF Core)	ORM tool to connect C# classes with SQL tables, manage migrations, and simplify SQL queries.
Dependency Injection (DI)	Used to inject services and repositories in controllers (to maintain clean architecture).
Newtonsoft.Json	Used to convert objects into JSON (example: saving cart items in order).

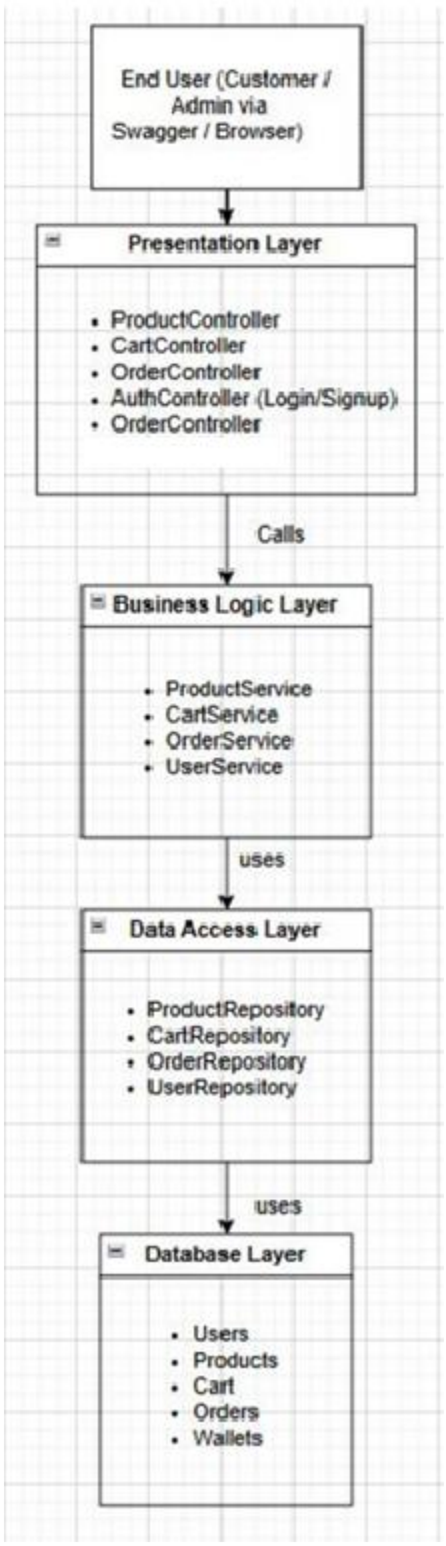
Data Flow Diagram


Merchant
Create , update or delete items
User
Pages with product names, price, description, add to cart option
Add To Cart
Show cart with total price, buy now option
Buy now
Get all information including username and password
Sign up
Login page Sign up page, Sign In, Login
Sign in
Authentication
Wallet Payment
Payment Cash on Delivery
COD
Order History

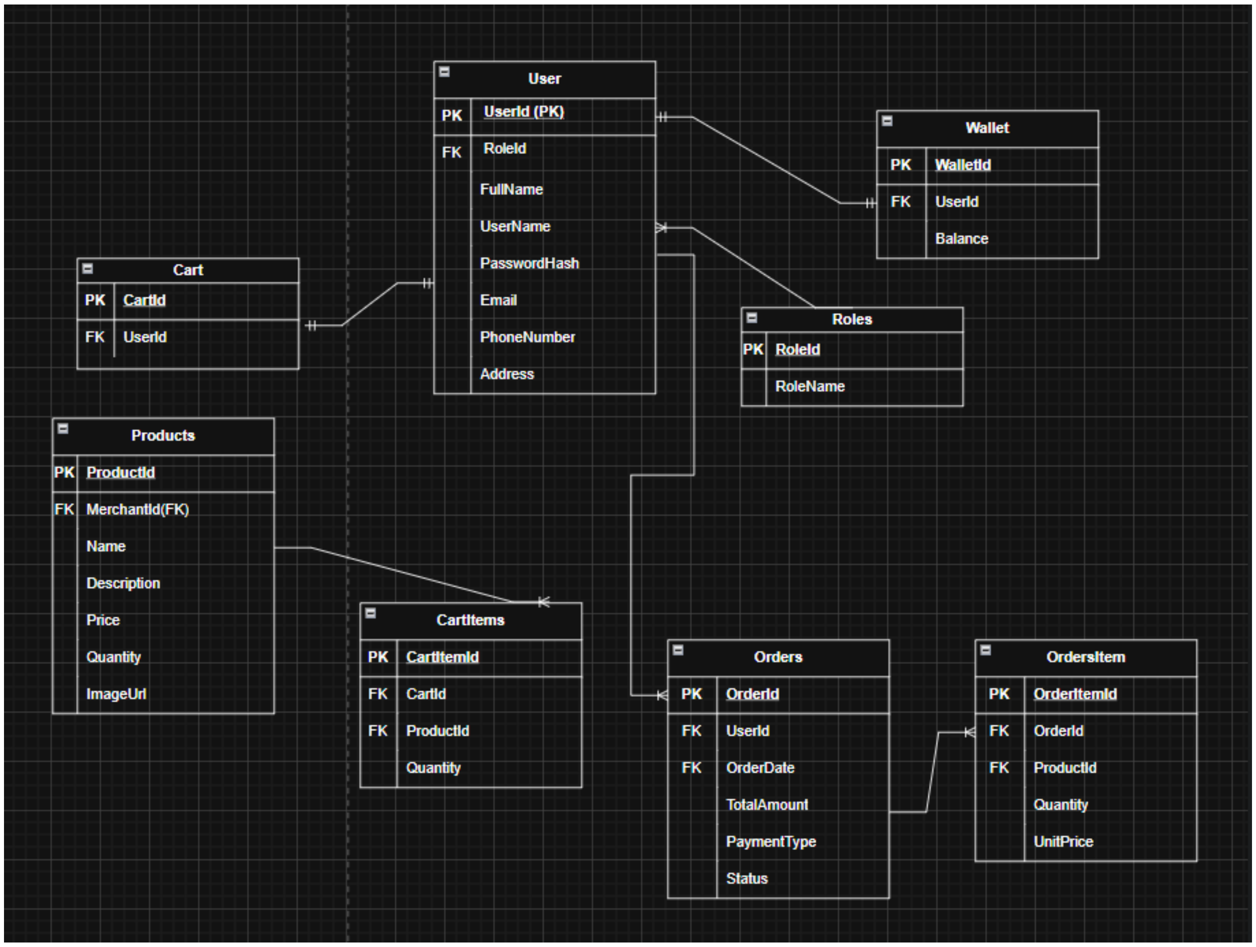
Sequential Diagram


User
Webpage
Database
Cart
AuthenticationService
GitHubLoginService
PaymentPage
WalletService (if wallet chosen)
Open webpage
Fetch all products
Return product list with name, price, description
Display all products with name, price, description
Add product to cart
Add selected product
Show cart total and buy option
Click buy option
Ask to login, sign in or sign up or github login
Choose sign in
Authenticate user credentials
Verify credentials
Return authentication result
Authentication success/failure status
Choose Sign Up
Get details
Save user details
Request GitHub login
Authenticate via GitHub
Verify/Create user from GitHub info
Navigate after successful login/ signup/github login
Show payment options (Cash on Delivery or Wallet)
Choose payment method
Process wallet payment

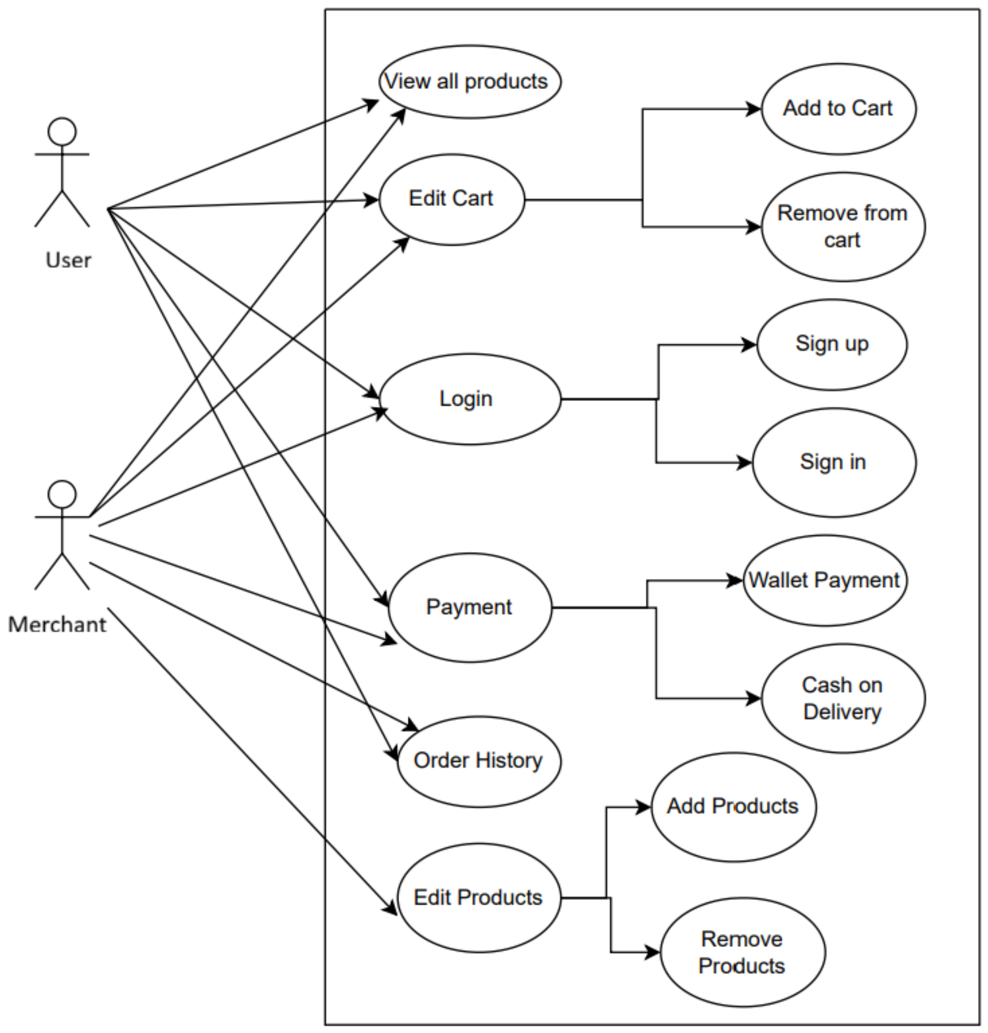
Architecture layer



ER Diagram


User
PK UserId (PK)
FK RoleId
FullName
UserName
PasswordHash
Email
PhoneNumber
Address
Wallet
PK WalletId
FK UserId
Balance
Cart
PK CartId
FK UserId
Roles
PK RoleId
RoleName
Products
PK ProductId
FK MerchantId(FK)
Name
Description
Price
Quantity
ImageUrl
CartItems
PK CartItemId
FK CartId
FK ProductId
Quantity
Orders
PK OrderId
FK UserId
FK OrderDate
TotalAmount
PaymentType
Status
OrdersItem
PK OrderItemId
FK OrderId
FK ProductId
Quantity
UnitPrice

Use Case Diagram



DataBase Schema

Role table

Column Name	Data Type	Description
RoleId	INT (PK)	Unique role ID
RoleName	VARCHAR(50)	e.g. “User”, “Merchant”

User Table

Column Name	Data Type	Description
UserId	INT (PK)	Unique ID
Name	VARCHAR(100)	Full name
Address	VARCHAR(250)	Address
Phone	VARCHAR(15)	Phone number
Email	VARCHAR(100)	Unique email
Username	VARCHAR(50)	Login username
PasswordHash	VARCHAR(MAX)	Hashed password
RoleId	INT (FK → Roles.RoleId)	Role type (User / Merchant)
WalletId	INT (FK → Wallets.WalletId)	Linked wallet

Wallet Table

Column Name	**Data Type**	**Description**
WalletId	INT (PK, Identity)	Unique wallet ID
UserId	INT (FK → Users.UserId)	Owner of the wallet
Balance	DECIMAL	Current wallet balance

Product Table

Column Name	**Data Type**	**Description**
ProductId	INT (PK)	Unique ID
ProductName	VARCHAR(100)	Product name
Description	VARCHAR(255)	Description
Price	DECIMAL(10,2)	Product price
Quantity	INT	Available stock
MerchantId	INT (FK → Users.UserId)	Merchant who added the product

Cart Table

Column Name	Data Type	Description
CartId	INT (PK)	Unique ID
UserId	INT (FK → Users.UserId)	Linked user
TotalAmount	DECIMAL	Total price of all items in the cart

Cart Items

Column Name	Data Type	Description
CartItemId	INT (PK, Identity)	Unique ID
CartId	INT (FK → Carts.CartId)`	Linked cart
ProductId	INT (FK → Products.ProductId)`	Product in the cart
Quantity	INT	Quantity selected
Total	DECIMAL	(Price × Quantity)

Order Table

Column Name	**Data Type**	**Description**
OrderId	INT (PK, Identity)`	Unique order ID
UserId	INT (FK → Users.UserId)`	Who made the order
TotalAmount	DECIMAL	Total cost
PaymentMethod	VARCHAR(50)	"Cash on Delivery" or "Wallet"
Status	VARCHAR(50)	e.g. "Pending", "Delivered"
OrderDate	DATETIME	When order was placed