Table of Content:

Problem Definition and Objectives

Overview of chosen technology for front end

Component Breakdown & API Design

Database Design

Backend Technology Stack Overview

Frontend - <http://localhost:4200/>

Backend - <http://localhost:5118>

Api testing ( Swagger UI ) - <http://localhost:5118/swagger>

Problem Definition and Objectives:

Project Name: ShopForHome is a popular store offering home décor products. Due to COVID-19, all offline shopping has stopped, and the store wants to move its operations to an online platform by creating a web application.

Overview of chosen technology for front end  
  
**Framework:** Angular (TypeScript-based, component-driven SPA framework)

* **Language:** TypeScript (with some HTML and CSS)
* **Build Tools:** Angular CLI (for scaffolding, building, serving, and testing)
* **Styling:** CSS (custom styles, possibly with Angular’s built-in style encapsulation)
* **Routing:** Angular Router (for navigation between views/pages)
* **Forms:** Reactive Forms (for robust form handling and validation)
* **HTTP:** Angular’s HttpClient (for API communication)
* **State Management:** Service-based (using Angular services for data and state)
* **Package Management:** npm (Node Package Manager)
* **Other:** Angular modules for organization, standalone components, and dependency injection.

**Component Breakdown & API Design**

**1. Major Frontend Components**

* **HomeComponent**  
  Displays hero section, categories, featured products, and site features.  
  *State:* Loads categories and featured products from backend.  
  *Routing:* Uses Angular Router for navigation (routerLink).  
  *UI:* Uses Angular template syntax (@for, @if), custom CSS.
* **ProductListComponent**  
  Handles product listing, filtering, sorting, pagination, and cart/wishlist actions.  
  *State:* Manages filter form, product response, categories.  
  *Routing:* Reads query params for filters/sorting.  
  *UI:* Displays products, filter controls, pagination.
* **ProductDetailComponent**  
  Shows details for a single product, add to cart/wishlist, reviews, etc.
* **CartComponent**  
  Displays cart items, allows quantity updates, removal, and checkout.
* **AuthComponent**  
  Handles login, registration, and authentication state.
* **Shared Components**  
  Header, Footer, Loader, Toast/Notification, etc.

**2. State Management**

* **Angular Services**
  + ProductService: Fetches products, categories, product details.
  + CartService: Manages cart and wishlist actions.
  + AuthService: Handles authentication, user state.
* **Reactive Forms**  
  Used for filters, login, registration, checkout forms.
* **Local State**  
  Component-level state for UI interactions (loading, error, etc.).

**3. Routing**

* **Angular Router**
  + /home: Home page
  + /products: Product listing (with query params for filters/sorting)
  + /products/:id: Product details
  + /cart: Cart
  + /login, /register: Auth
  + Guards for protected routes (cart, wishlist, checkout).

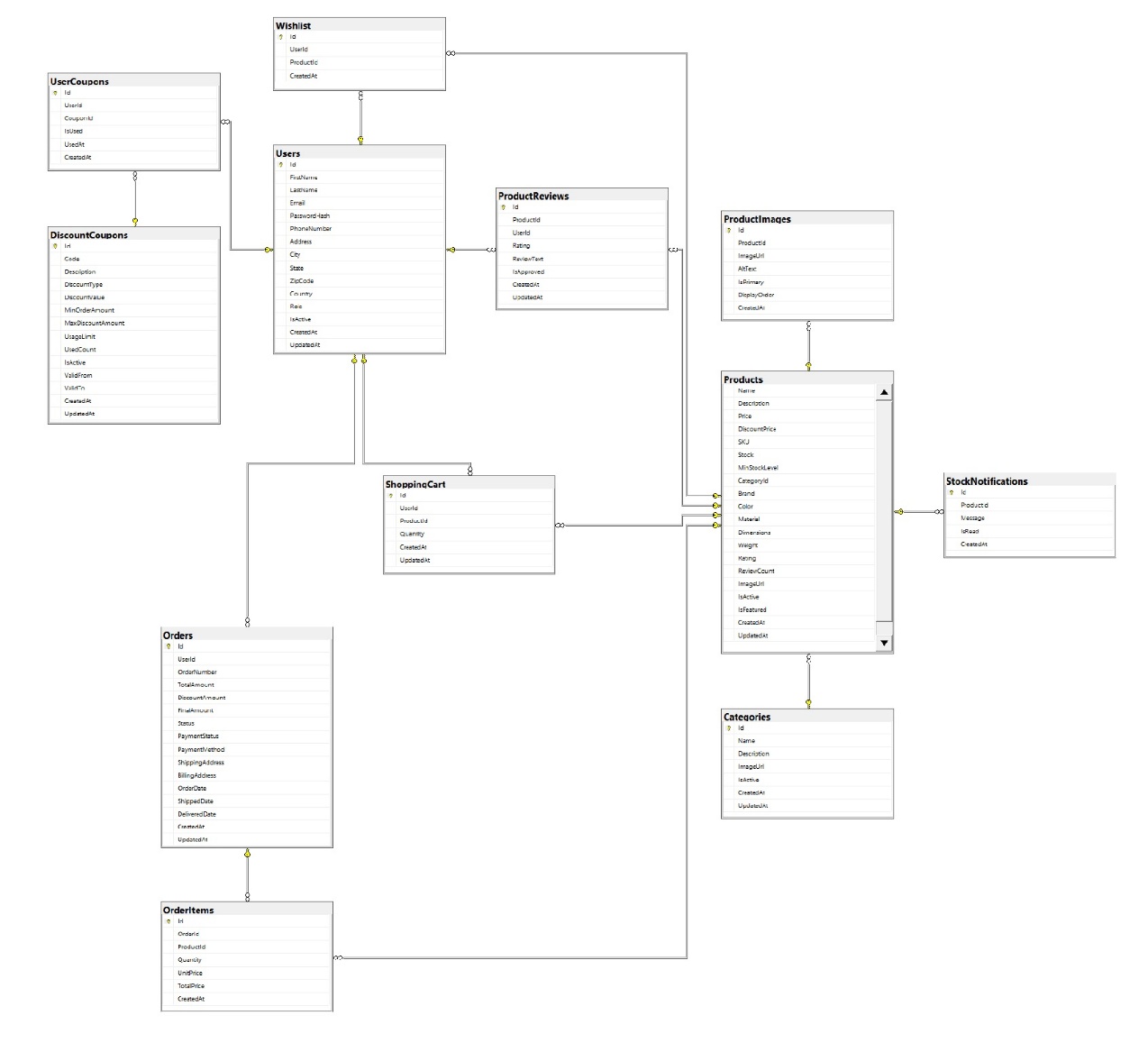
**4. API Design & Endpoints**

* **RESTful Endpoints (Example):**
  + GET /api/products  
    Query params: searchTerm, categoryId, minPrice, maxPrice, sortBy, sortOrder, page, pageSize, etc.
  + GET /api/products/:id  
    Fetch product details.
  + GET /api/categories  
    List all categories.
  + POST /api/cart/add  
    Add product to cart.
  + POST /api/wishlist/toggle  
    Add/remove product from wishlist.
  + POST /api/auth/login  
    User login.
  + POST /api/auth/register  
    User registration.
  + GET /api/cart  
    Get current cart items.

**5. Authentication Mechanism**

* **JWT (JSON Web Token)**
  + On login/register, backend returns JWT.
  + Token stored in browser (localStorage or cookies).
  + Token sent in Authorization: Bearer <token> header for protected endpoints.
  + Angular AuthService manages login state, token storage, and guards.

Database Design



The database for ShopForHome is designed to support an e-commerce platform with the following main entities:

**1. Products**

* Stores product details: name, description, price, effective price, images, category, brand, color, stock, rating, review count, sale status, featured flag.

**2. Categories**

* Organizes products into categories (e.g., Furniture, Decor).
* Each category has a name, image, and product count.

**3. Users**

* Stores user information: name, email, password hash, address, phone, and user role (customer/admin).

**4. Cart**

* Temporary storage for products a user intends to buy.
* Linked to user, contains product IDs and quantities.

**5. Wishlist**

* Products a user has marked for future interest.
* Linked to user and product.

**6. Orders**

* Stores completed purchases.
* Includes user, products, quantities, prices, order status, payment info, shipping address.

**7. Reviews**

* User-generated product reviews.
* Includes rating, comment, user, product, and date.

**8. Authentication**

* Stores tokens or session info for user login/logout.

DATABASE SCHEMA

-- ShopForHome Database Schema

-- E-commerce application for home décor products

USE master;

GO

-- Create database if it doesn't exist

IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = 'ShopForHomeDB')

BEGIN

CREATE DATABASE ShopForHomeDB;

END

GO

USE ShopForHomeDB;

GO

-- Users table (for both regular users and admins)

CREATE TABLE Users (

Id INT IDENTITY(1,1) PRIMARY KEY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

Email NVARCHAR(100) UNIQUE NOT NULL,

PasswordHash NVARCHAR(255) NOT NULL,

PhoneNumber NVARCHAR(15),

Address NVARCHAR(500),

City NVARCHAR(50),

State NVARCHAR(50),

ZipCode NVARCHAR(10),

Country NVARCHAR(50),

Role NVARCHAR(20) NOT NULL DEFAULT 'User', -- 'User' or 'Admin'

IsActive BIT NOT NULL DEFAULT 1,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()

);

-- Categories table

CREATE TABLE Categories (

Id INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(100) NOT NULL UNIQUE,

Description NVARCHAR(500),

ImageUrl NVARCHAR(255),

IsActive BIT NOT NULL DEFAULT 1,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()

);

-- Products table

CREATE TABLE Products (

Id INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(200) NOT NULL,

Description NVARCHAR(1000),

Price DECIMAL(10,2) NOT NULL,

DiscountPrice DECIMAL(10,2),

SKU NVARCHAR(50) UNIQUE NOT NULL,

Stock INT NOT NULL DEFAULT 0,

MinStockLevel INT NOT NULL DEFAULT 10,

CategoryId INT NOT NULL,

Brand NVARCHAR(100),

Color NVARCHAR(50),

Material NVARCHAR(100),

Dimensions NVARCHAR(100),

Weight DECIMAL(8,2),

Rating DECIMAL(3,2) DEFAULT 0,

ReviewCount INT DEFAULT 0,

ImageUrl NVARCHAR(255),

IsActive BIT NOT NULL DEFAULT 1,

IsFeatured BIT NOT NULL DEFAULT 0,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (CategoryId) REFERENCES Categories(Id)

);

-- Product Images table (for multiple images per product)

CREATE TABLE ProductImages (

Id INT IDENTITY(1,1) PRIMARY KEY,

ProductId INT NOT NULL,

ImageUrl NVARCHAR(255) NOT NULL,

AltText NVARCHAR(200),

IsPrimary BIT NOT NULL DEFAULT 0,

DisplayOrder INT NOT NULL DEFAULT 0,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE

);

-- Shopping Cart table

CREATE TABLE ShoppingCart (

Id INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL,

ProductId INT NOT NULL,

Quantity INT NOT NULL DEFAULT 1,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,

FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE,

UNIQUE(UserId, ProductId)

);

-- Wishlist table

CREATE TABLE Wishlist (

Id INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL,

ProductId INT NOT NULL,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,

FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE,

UNIQUE(UserId, ProductId)

);

-- Orders table

CREATE TABLE Orders (

Id INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL,

OrderNumber NVARCHAR(50) UNIQUE NOT NULL,

TotalAmount DECIMAL(10,2) NOT NULL,

DiscountAmount DECIMAL(10,2) DEFAULT 0,

FinalAmount DECIMAL(10,2) NOT NULL,

Status NVARCHAR(50) NOT NULL DEFAULT 'Pending', -- Pending, Confirmed, Shipped, Delivered, Cancelled

PaymentStatus NVARCHAR(50) NOT NULL DEFAULT 'Pending', -- Pending, Paid, Failed, Refunded

PaymentMethod NVARCHAR(50),

ShippingAddress NVARCHAR(500) NOT NULL,

BillingAddress NVARCHAR(500),

OrderDate DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

ShippedDate DATETIME2,

DeliveredDate DATETIME2,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (UserId) REFERENCES Users(Id)

);

-- Order Items table

CREATE TABLE OrderItems (

Id INT IDENTITY(1,1) PRIMARY KEY,

OrderId INT NOT NULL,

ProductId INT NOT NULL,

Quantity INT NOT NULL,

UnitPrice DECIMAL(10,2) NOT NULL,

TotalPrice DECIMAL(10,2) NOT NULL,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (OrderId) REFERENCES Orders(Id) ON DELETE CASCADE,

FOREIGN KEY (ProductId) REFERENCES Products(Id)

);

-- Discount Coupons table

CREATE TABLE DiscountCoupons (

Id INT IDENTITY(1,1) PRIMARY KEY,

Code NVARCHAR(50) UNIQUE NOT NULL,

Description NVARCHAR(200),

DiscountType NVARCHAR(20) NOT NULL, -- 'Percentage' or 'FixedAmount'

DiscountValue DECIMAL(10,2) NOT NULL,

MinOrderAmount DECIMAL(10,2) DEFAULT 0,

MaxDiscountAmount DECIMAL(10,2),

UsageLimit INT,

UsedCount INT DEFAULT 0,

IsActive BIT NOT NULL DEFAULT 1,

ValidFrom DATETIME2 NOT NULL,

ValidTo DATETIME2 NOT NULL,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()

);

-- User Coupons table (for assigning coupons to specific users)

CREATE TABLE UserCoupons (

Id INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL,

CouponId INT NOT NULL,

IsUsed BIT NOT NULL DEFAULT 0,

UsedAt DATETIME2,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,

FOREIGN KEY (CouponId) REFERENCES DiscountCoupons(Id) ON DELETE CASCADE,

UNIQUE(UserId, CouponId)

);

-- Product Reviews table

CREATE TABLE ProductReviews (

Id INT IDENTITY(1,1) PRIMARY KEY,

ProductId INT NOT NULL,

UserId INT NOT NULL,

Rating INT NOT NULL CHECK (Rating >= 1 AND Rating <= 5),

ReviewText NVARCHAR(1000),

IsApproved BIT NOT NULL DEFAULT 0,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE,

FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,

UNIQUE(ProductId, UserId)

);

-- Stock Notifications table (for admin notifications when stock is low)

CREATE TABLE StockNotifications (

Id INT IDENTITY(1,1) PRIMARY KEY,

ProductId INT NOT NULL,

Message NVARCHAR(500) NOT NULL,

IsRead BIT NOT NULL DEFAULT 0,

CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),

FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE

);

-- Insert default categories

INSERT INTO Categories (Name, Description, ImageUrl) VALUES

('Furniture', 'Home furniture including sofas, chairs, tables, and storage solutions', '/assets/images/categories/furniture.jpg'),

('Home Décor', 'Decorative items to beautify your home including wall art, vases, and ornaments', '/assets/images/categories/home-decor.jpg'),

('Lighting', 'Indoor and outdoor lighting solutions including lamps, chandeliers, and LED lights', '/assets/images/categories/lighting.jpg'),

('Kitchen & Dining', 'Kitchen appliances, cookware, and dining accessories', '/assets/images/categories/kitchen-dining.jpg'),

('Bedroom', 'Bedroom furniture and accessories including beds, mattresses, and bedding', '/assets/images/categories/bedroom.jpg'),

('Bathroom', 'Bathroom fixtures, accessories, and storage solutions', '/assets/images/categories/bathroom.jpg');

-- Insert default admin user (password: Admin@123)

INSERT INTO Users (FirstName, LastName, Email, PasswordHash, Role, PhoneNumber, Address, City, State, Country) VALUES

('Admin', 'User', 'admin@shopforhome.com', 'AQAAAAEAACcQAAAAEJ5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q5Q==', 'Admin', '+1234567890', '123 Admin Street', 'Admin City', 'Admin State', 'USA');

-- Create indexes for better performance

CREATE INDEX IX\_Products\_CategoryId ON Products(CategoryId);

CREATE INDEX IX\_Products\_Price ON Products(Price);

CREATE INDEX IX\_Products\_Rating ON Products(Rating);

CREATE INDEX IX\_Products\_IsActive ON Products(IsActive);

CREATE INDEX IX\_Products\_IsFeatured ON Products(IsFeatured);

CREATE INDEX IX\_ShoppingCart\_UserId ON ShoppingCart(UserId);

CREATE INDEX IX\_Wishlist\_UserId ON Wishlist(UserId);

CREATE INDEX IX\_Orders\_UserId ON Orders(UserId);

CREATE INDEX IX\_Orders\_OrderDate ON Orders(OrderDate);

CREATE INDEX IX\_OrderItems\_OrderId ON OrderItems(OrderId);

CREATE INDEX IX\_ProductReviews\_ProductId ON ProductReviews(ProductId);

CREATE INDEX IX\_UserCoupons\_UserId ON UserCoupons(UserId);

CREATE INDEX IX\_DiscountCoupons\_Code ON DiscountCoupons(Code);

CREATE INDEX IX\_DiscountCoupons\_ValidFrom\_ValidTo ON DiscountCoupons(ValidFrom, ValidTo);

-- Create triggers for updating product ratings

GO

CREATE TRIGGER TR\_UpdateProductRating

ON ProductReviews

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

SET NOCOUNT ON;

-- Update ratings for affected products

UPDATE p

SET Rating = ISNULL(r.AvgRating, 0),

ReviewCount = ISNULL(r.ReviewCount, 0),

UpdatedAt = GETUTCDATE()

FROM Products p

LEFT JOIN (

SELECT

ProductId,

AVG(CAST(Rating AS DECIMAL(3,2))) as AvgRating,

COUNT(\*) as ReviewCount

FROM ProductReviews

WHERE IsApproved = 1

GROUP BY ProductId

) r ON p.Id = r.ProductId

WHERE p.Id IN (

SELECT DISTINCT ProductId FROM inserted

UNION

SELECT DISTINCT ProductId FROM deleted

);

END;

GO

-- Create trigger for stock notifications

CREATE TRIGGER TR\_StockNotification

ON Products

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StockNotifications (ProductId, Message)

SELECT

i.Id,

'Product "' + i.Name + '" stock level is low (' + CAST(i.Stock AS NVARCHAR(10)) + ' remaining). Minimum stock level: ' + CAST(i.MinStockLevel AS NVARCHAR(10))

FROM inserted i

INNER JOIN deleted d ON i.Id = d.Id

WHERE i.Stock <= i.MinStockLevel

AND d.Stock > d.MinStockLevel; -- Only trigger when stock drops below minimum

END;

GO

PRINT 'ShopForHome database schema created successfully!';

**Backend Technology Stack Overview**

* **Language & Framework:**  
  C# with ASP.NET Core (see Program.cs, .csproj, Controllers, Services)
* **Project Type:**  
  ASP.NET Core Web API (ShopForHome.API.csproj)
* **Database:**  
  Likely SQL Server (based on Microsoft.EntityFrameworkCore.SqlServer.dll and ShopForHomeDbContext.cs)
* **ORM:**  
  Entity Framework Core (see Data/ShopForHomeDbContext.cs, Migrations/)
* **Authentication:**  
  JWT Bearer Authentication (Microsoft.AspNetCore.Authentication.JwtBearer.dll, AuthController.cs, AuthService.cs)
* **API Structure:**  
  RESTful controllers (Controllers/ folder: ProductsController, CategoriesController, CartController, etc.)
* **DTOs:**  
  Data Transfer Objects for API requests/responses (DTOs/ folder)
* **Services:**  
  Business logic separated into services (Services/ folder)
* **Configuration:**  
  App settings in appsettings.json and appsettings.Development.json
* **Migrations:**  
  Database migrations managed via Entity Framework (Migrations/ folder)

**Summary:**  
Your backend is a modern ASP.NET Core Web API project using Entity Framework Core for data access, SQL Server as the database, JWT for authentication, and a clean separation of concerns with controllers, services, DTOs, and models.

Screenshot

