**CAPSTONE PROJECT**

**FINAL REPORT**

**ShopForHome – E-Commerce Web Application**

**(.NET + ANGULAR)**

**SUBMITTED BY**

DEEPIKA S M

WIPRO NGA - .Net Full Stack Angular - FY26 – C2

**UNDER THE GUIDANCE OF**

JYOTI S PATIL

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to my mentor Mr.Harshavardhan Shandilya, faculties – Niti Dwivedi and Jyoti S Patil for their continuous guidance and support throughout the completion of this capstone project “ShopForHome-Web Application”.

This project would not have been possible without the invaluable learning resources, documentation, and training provided by **“Wipro Pre-Skill Training”** through **“Great Learning Platform”**. I am especially thankful to my instructors for their constructive feedback, which helped me refine the technical design, coding practices, and testing approach.

I also wish to acknowledge the support of my colleagues and friends who encouraged me and provided helpful insights during the development of both the backend and frontend modules.

Finally, I am grateful for the opportunity to apply the skills acquired during the course in a real-world project, which has greatly enhanced my confidence and understanding of **ASP.NET Core, Angular, SQL, and software testing practices**.

|  |  |  |
| --- | --- | --- |
| **S. No** | **CONTENTS** | **Page No.** |
| **1** | **ABSTRACT** | **1** |
| **2** | **PROBLEM STATEMENT** | **2** |
| **3** | **PROBLEM DEFINITION & OBJECTIVES** | **3** |
| **4** | **INTRODUCTION** | **7** |
| **5** | **FRONTEND AND BACKEND ARCHITECTURE** | **8** |
| **6** | **SYSTEM DESIGN** | **9** |
| **7** | **SQL DATABASE SCHEMA**   * **ER Description** * **Core Table and Columns** * **Database Schema** | **10** |
| **8** | **PROJECT OVERVIEW DIAGRAM** | **19** |
| **9** | **USE CASE DIAGRAM** | **20** |
| **10** | **CLASS DIAGRAM** | **21** |
| **11** | **USER SEQUENCE DIAGRAM** | **22** |
| **12** | **ADMIN SEQUENCE DIAGRAM** | **23** |
| **13** | **ER DIAGRAM** | **24** |
| **14** | **FRONTEND CODE SCREENSHOTS** | **25** |
| **15** | **BACKEND CODE SCREENSHOTS** | **25** |
| **16** | **FRONTEND TESTING SCREENSHOTS** | **26** |
| **17** | **BACKEND TESTING SCREENSHOTS** | **26** |
| **18** | **HOME PAGE** | **27** |
| **19** | **LOGIN PAGE** | **27** |
| **20** | **REGISTER PAGE** | **28** |
| **21** | **PRODUCTS PAGE** | **28** |
| **22** | **WISHLIST PAGE** | **29** |
| **23** | **CART PAGE** | **29** |
| **24** | **CHECKOUT PAGE** | **30** |
| **25** | **ADMIN-DASHBOARD,PRODUCTS,ORDERS, PAGE** | **31** |
| **26** | **ADMIN-COUPON PAGE** | **32** |
| **27** | **CONCLUSION** | **33** |

**ABSTRACT**

The ShopForHome Capstone Project is a full-stack web application developed to transform a traditional home décor retail store into an online e-commerce platform in response to the challenges posed by COVID-19. The system is designed with two primary roles—User and Admin—to ensure smooth online shopping and efficient store management. Users can register, log in, browse products by category, filter items, manage shopping carts, create wishlists, and apply discount coupons. On the other hand, Admins are provided with advanced functionalities such as user management, product CRUD operations, bulk product upload via CSV, stock monitoring, discount coupon management, and sales report generation.

The application architecture leverages Angular for the front end, offering a dynamic and responsive user interface, and ASP.NET Core MVC for the back end, which manages business logic, APIs, and database operations. Entity Framework is used for ORM-based database interaction, while Swagger is integrated for API testing and documentation. The development was structured across multiple sprints following Agile methodology, ensuring iterative progress on database design, CRUD operations, shopping cart functionality, search and filtering features, and admin reporting tools.

This project demonstrates a scalable and modern e-commerce solution that bridges the gap between offline and online retail, providing customers with a seamless shopping experience and administrators with robust management capabilities.

**PROBLEM DEFINITION AND OBJECTIVES**

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.

There are two types of users in this application:

**1. User**

**2. Admin**

**OBJECTIVES**

**1. For Users (Customers):**

* Provide secure login, registration, and profile management.
* Enable browsing and filtering of products by category, price, and rating.
* Support shopping cart management with quantity adjustments.
* Offer wishlist functionality for saving products for later purchase.
* Integrate discount coupon features for promotions.

**2. For Admins:**

* Implement user management with full CRUD operations.
* Provide product management with CRUD and bulk CSV upload support.
* Monitor stock levels and trigger notifications for low inventory.
* Create, assign, and manage discount coupons for specific users.
* Generate sales reports for business analysis.

**3. System-Level Objectives:**

* Use Angular for a responsive and user-friendly frontend.
* Use ASP.NET Core MVC & Web API for backend business logic and REST services.
* Employ SQL Server with Entity Framework for efficient database operations.
* Ensure security using authentication (JWT tokens) and role-based access control.
* Provide API documentation and testing via Swagger.

**INTRODUCTION**

The rapid growth of e-commerce has transformed the way people shop and businesses operate. Customers now expect **convenience, personalization, and efficiency**, while businesses demand platforms that are **scalable, secure, and easy to manage**. In this context, the **ShopForHome** project has been developed as a comprehensive e-commerce solution that bridges the gap between customer expectations and business requirements.

The system provides a **user-friendly Angular-based frontend** for seamless navigation, product exploration, cart management, and wishlist functionality. On the backend, it leverages the **ASP.NET Core Web API** with **Entity Framework Core** and **SQL Server** to ensure robust data management, scalability, and secure operations. Key features include **user authentication with JWT tokens, role-based access (User/Admin), coupon and discount management, order placement, and sales reporting**.

For administrators, the platform offers a **dedicated dashboard** to manage products, track stock, generate sales reports, and upload bulk product data via CSV. For customers, it ensures a smooth shopping journey, from browsing products to placing orders and applying discounts. Supporting tools such as **Swagger for API documentation** and **Postman for testing** make development and maintenance more efficient.

By combining **modern web technologies, secure authentication mechanisms, and a modular architecture**, ShopForHome provides a reliable and scalable solution that can adapt to evolving business and user needs.

**FRONTEND AND BACKEND ARCHITECTURE**

**TECHNOLOGY STACK**

* + Frontend: Angular
  + Backend: ASP.NET Core MVC & Web API
  + Database: SQL Server with Entity Framework Core
  + API Documentation & Testing: Swagger (Swashbuckle)

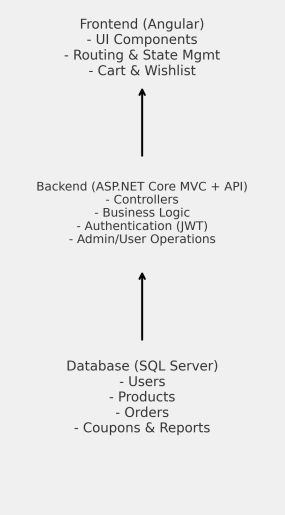
**Supporting Tools & Libraries**

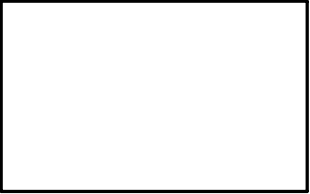
* **Swagger** for API testing and documentation.
* **CSV Upload** feature for bulk product insertion by Admin.
* **JWT Authentication** for secure access control.

**IDE / DEVELOPMENT TOOLS:**

* Visual Studio Code (for frontend development)
* SQL Server Management Studio (SSMS) (for database management)
* Postman / Swagger UI (for API testing)
* Git & GitHub (for version control)

**SYSTEM DESIGN**

****



**COMPONENT BREAKDOWN & API DESIGN**

**Frontend Components (Angular)**

The frontend of *ShopForHome* is implemented in **Angular**, structured as reusable components to ensure modularity and maintainability.

* **State Management**
  + Managed using Angular Services.
  + Handles shared data such as user authentication state, cart items, and wishlist.
  + Ensures consistent state across components without redundant API calls.
* **Routing**
  + Angular Router is used for navigation.
  + Routes include:
    - /login – User/Admin login page
    - /register – New user registration
    - /products – Product listing by category
    - /cart – Shopping cart management
    - /wishlist – Wishlist management
    - /admin – Admin dashboard (secured, role-based access)
* **UI Components**
  + **Header**  – Common navigation elements across all pages.
  + **Product Listing Component** – Displays products with filtering and sorting options.
  + **Product Detail Component** – Shows detailed information about a selected product.
  + **Cart Component** – Allows users to add, update, or remove items from the shopping cart.
  + **Wishlist Component** – Stores products for later purchase.
  + **Admin Components** – Manage users, products, coupons, and stock reports.

**Backend API Design (ASP.NET Core Web API)**

The backend provides **RESTful API endpoints** for communication between the Angular frontend and SQL Server database. Authentication is secured using **JWT (JSON Web Tokens)**.

* **Authentication Endpoints**
  + **POST** /api/auth/login → Authenticate user/admin and issue JWT token.
  + **POST** /api/auth/register → Register a new user.
  + **POST** /api/auth/logout → Invalidate token (handled client-side).
* **User Endpoints**
  + **GET** /api/users → Retrieve all users (Admin only).
  + **GET** /api/users/{id} → Get user details.
  + **PUT** /api/users/{id} → Update user details.
  + **DELETE** /api/users/{id} → Delete user account.
* **Product Endpoints**
  + **GET** /api/products → Get all products with filters (category, price, rating).
  + **GET** /api/products/{id} → Get product details by ID.
  + **POST** /api/products → Add a new product (Admin only).
  + **PUT** /api/products/{id} → Update product details (Admin only).
  + **DELETE** /api/products/{id} → Delete product (Admin only).
  + **POST** /api/products/upload-csv → Bulk product upload via CSV.
* **Cart Endpoints**
  + **GET** /api/cart/{userId} → Retrieve items in a user’s cart.
  + **POST** /api/cart/add → Add product to cart.
  + **PUT** /api/cart/update/{cartItemId} → Update quantity.
  + **DELETE** /api/cart/remove/{cartItemId} → Remove item from cart.
* **Wishlist Endpoints**
  + **GET** /api/wishlist/{userId} → Get wishlist items.
  + **POST** /api/wishlist/add → Add product to wishlist.
  + **DELETE** /api/wishlist/remove/{id} → Remove item from wishlist.
* **Coupon & Discount Endpoints**
  + **GET** /api/coupons → Get available coupons.
  + **POST** /api/coupons → Create new coupon (Admin only).
  + **POST** /api/coupons/assign → Assign coupon to specific users (Admin only).
* **Reports & Stock Management**
  + **GET** /api/reports/sales?startDate=&endDate= → Generate sales report.
  + **GET** /api/products/stock → View stock levels and low-stock alerts.

**Authentication Mechanism**

* **JWT (JSON Web Token)** is used for authentication and authorization.
* On login, the server generates a token containing user role (User/Admin).
* The token must be included in the **Authorization header** of subsequent requests:

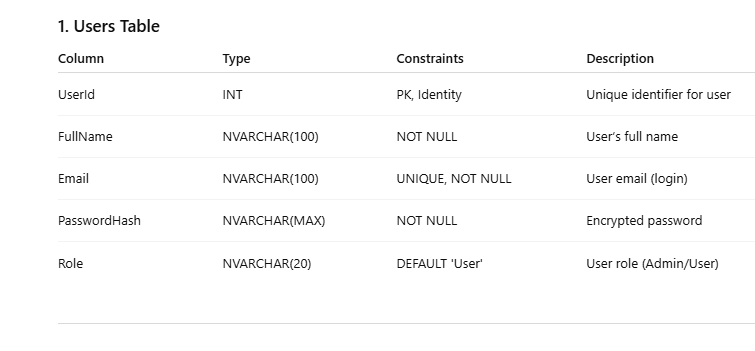
**SQL DATABASE SCHEMA**

**ER DESCRIPTION:**

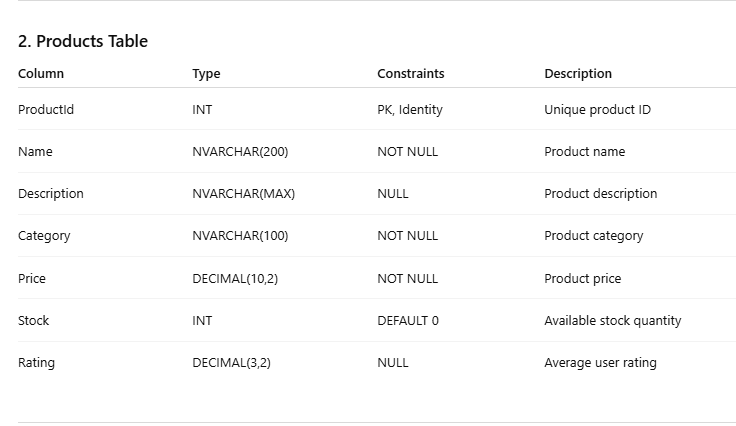
The ShopForHome database is designed using an Entity–Relationship (ER) model that maps real-world entities (users, products, orders, etc.) into relational tables with defined relationships. The major entities and their relationships are:

* Users → Orders (1:N)
  + A single user can place multiple orders.
  + Each order belongs to exactly one user.
  + Enforced by Orders.UserId → Users.UserId.
* Orders → OrderItems (1:N)
  + Each order may contain multiple items (products).
  + Each item in an order refers to one product.
  + Enforced by OrderItems.OrderId → Orders.OrderId.
* Products → OrderItems (1:N)
  + A product can appear in multiple order items.
  + Each order item refers to a single product.
  + Enforced by OrderItems.ProductId → Products.ProductId.
* Users → Cart (1:N)
  + A user can have multiple products in their cart.
  + Each cart item belongs to exactly one user.
  + Enforced by Cart.UserId → Users.UserId.
* Products → Cart (1:N)
  + A product can be present in multiple users’ carts.
  + Each cart item refers to one product.
  + Enforced by Cart.ProductId → Products.ProductId.
* Users → Wishlist (1:N)
  + A user can add multiple products to their wishlist.
  + Each wishlist entry belongs to one user.
  + Enforced by Wishlist.UserId → Users.UserId.
* Products → Wishlist (1:N)
  + A product can be wishlisted by multiple users.
  + Each wishlist entry refers to one product.
  + Enforced by Wishlist.ProductId → Products.ProductId.
* Coupons (Independent Entity)
  + Coupons are standalone entities with attributes like Code, Discount, and ExpiryDate.
  + They can be applied during checkout to reduce order total.
  + Relation to orders is maintained at the application logic level (not hard foreign key in DB).

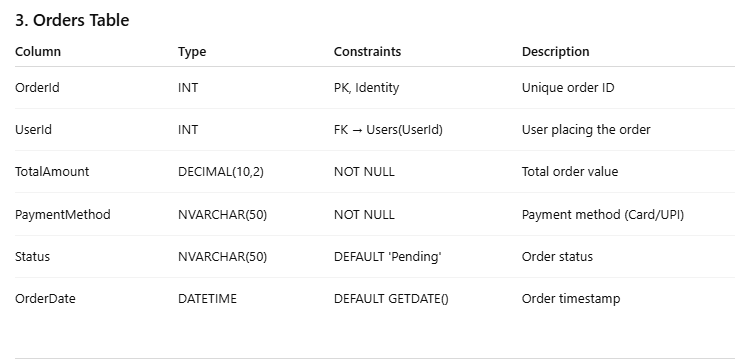
**DATABASE TABLES:**

****

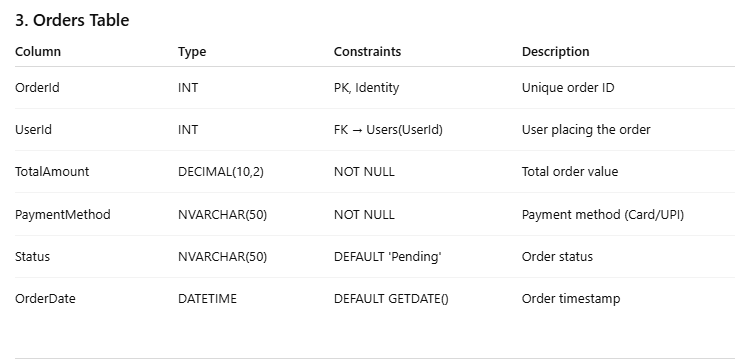
**PRODUCTS TABLE :**

****

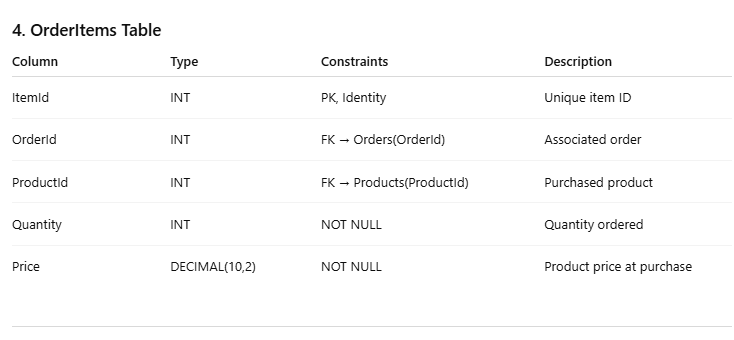
**ORDERS TABLE :**

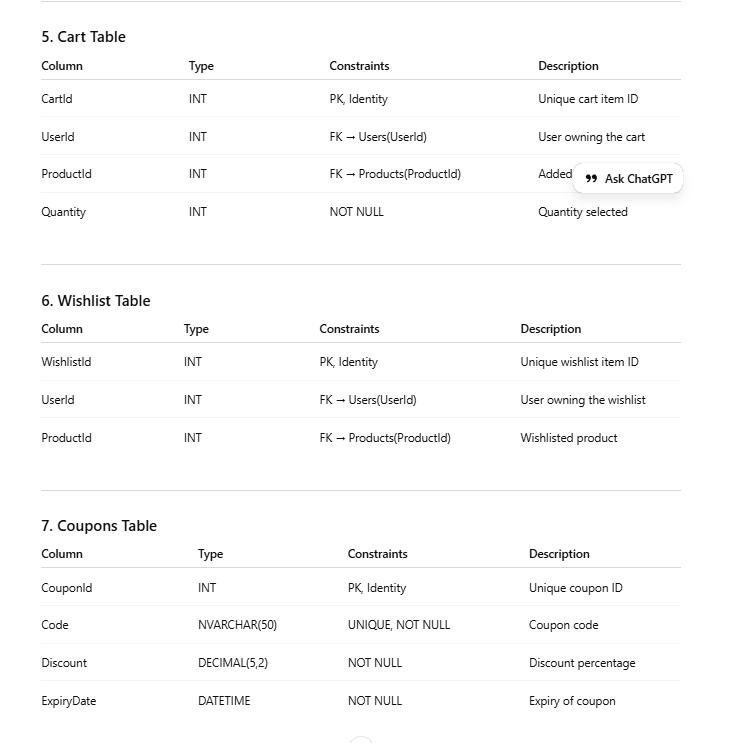
****

**ORDERS TABLE :**

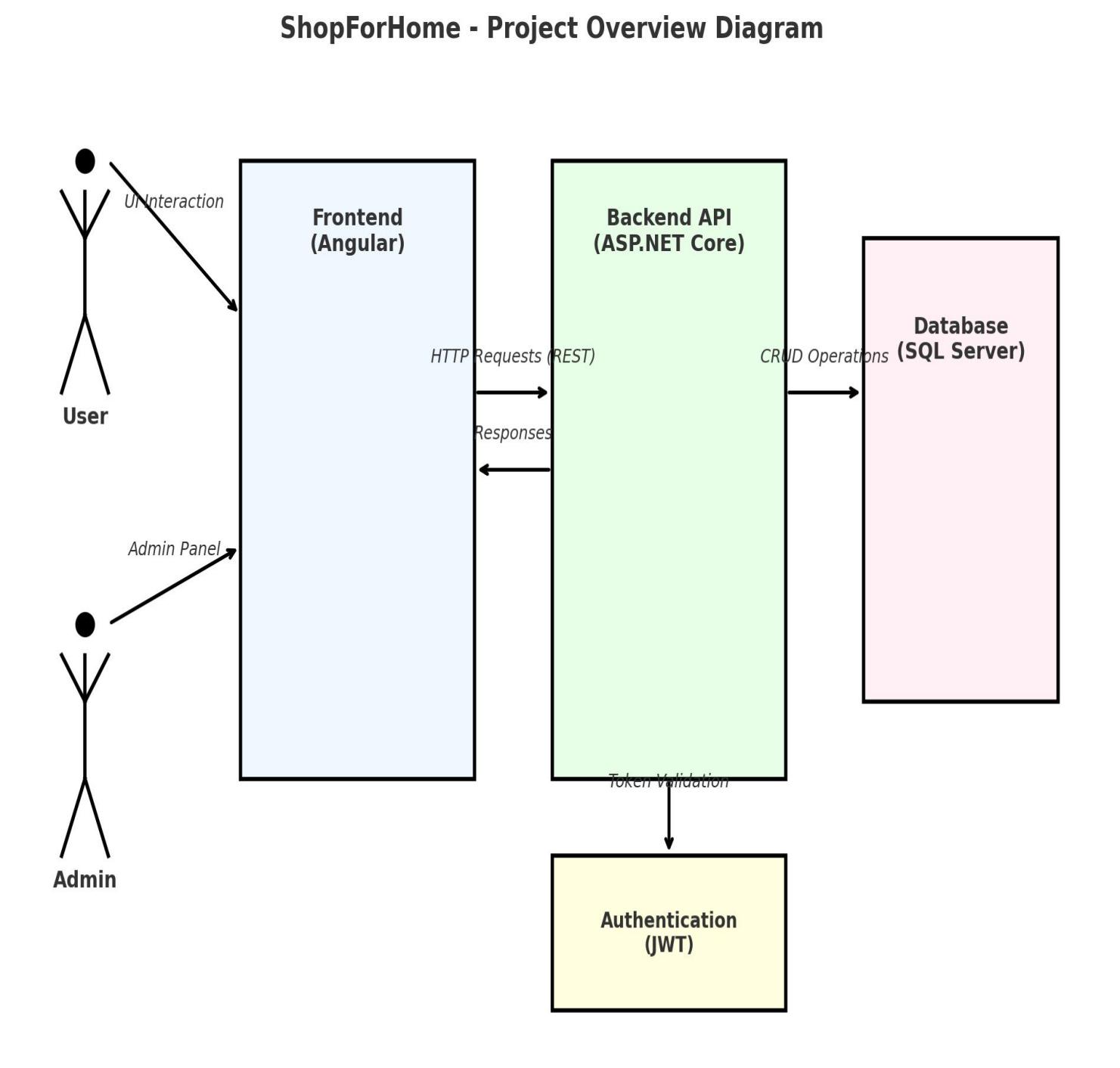
****

**ORDERSITEMS TABLE:**

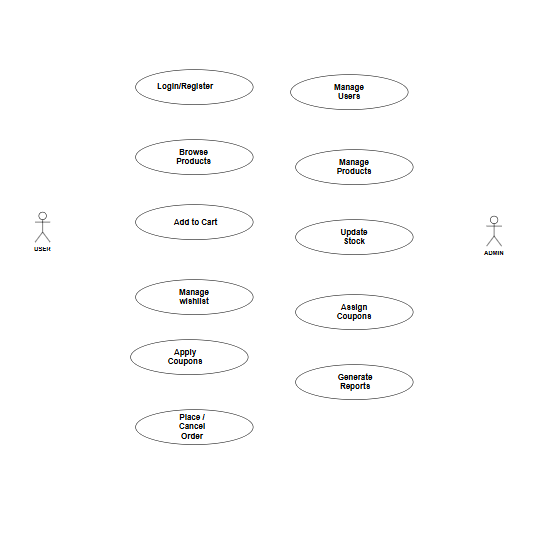
****

****

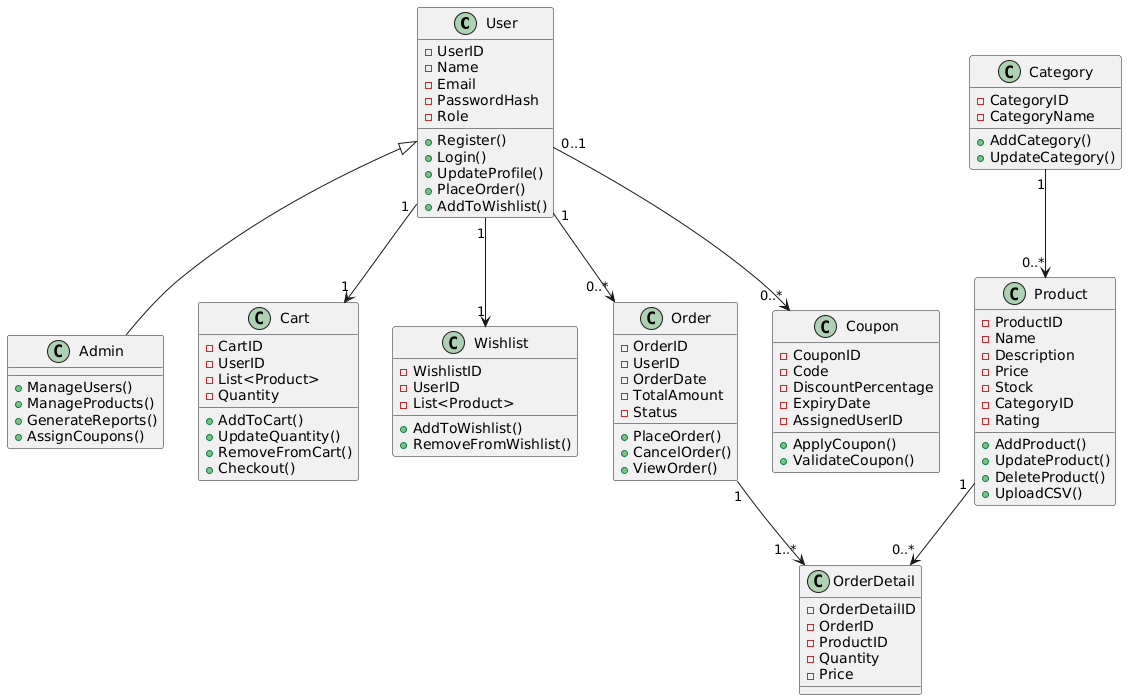
**PROJECT OVERVIEW DIAGRAM**



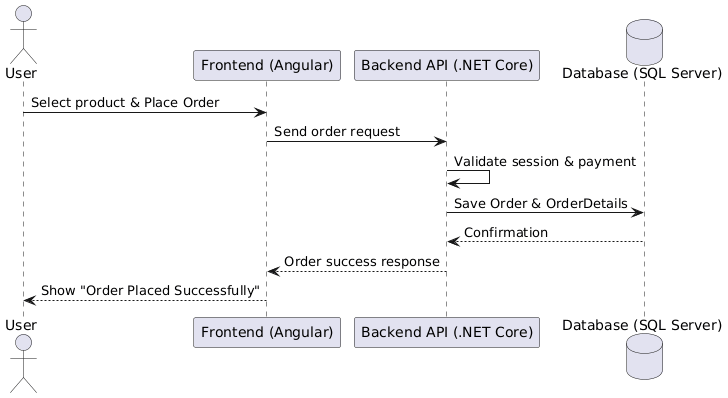
**USE CASE DIAGRAM**



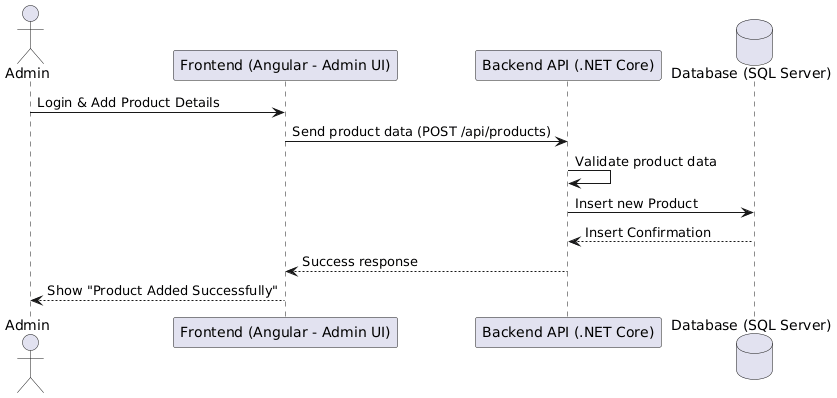
**CLASS DIAGRAM**

****

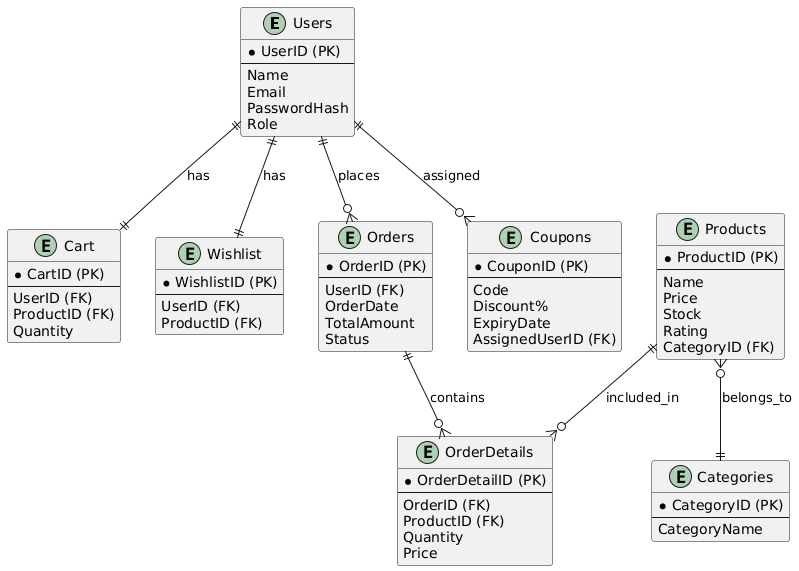
**USER SEQUENCE DIAGRAM**



**ADMIN SEQUENCE DIAGRAM**

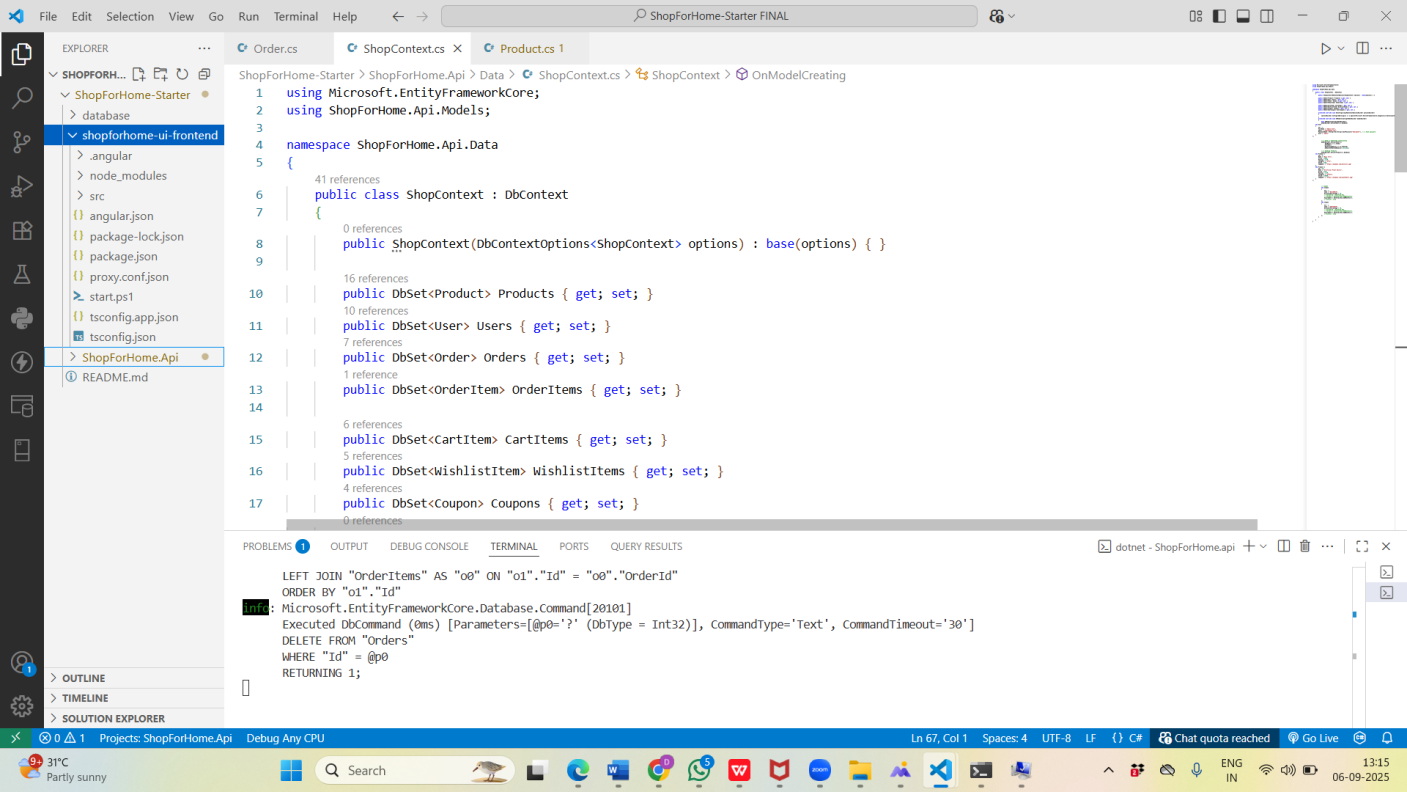


**ER DIAGRAM**

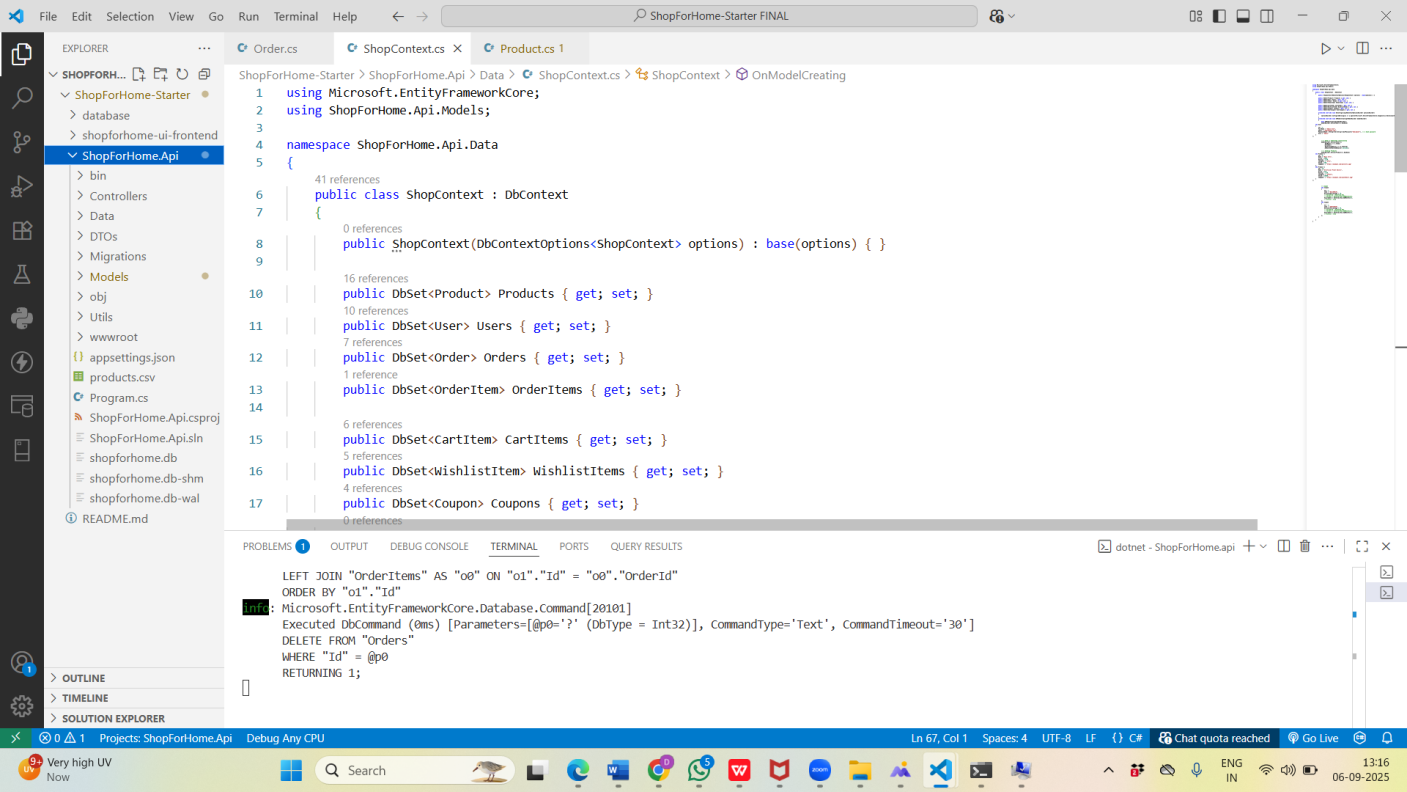


**PROJECT SCREENSHOTS**

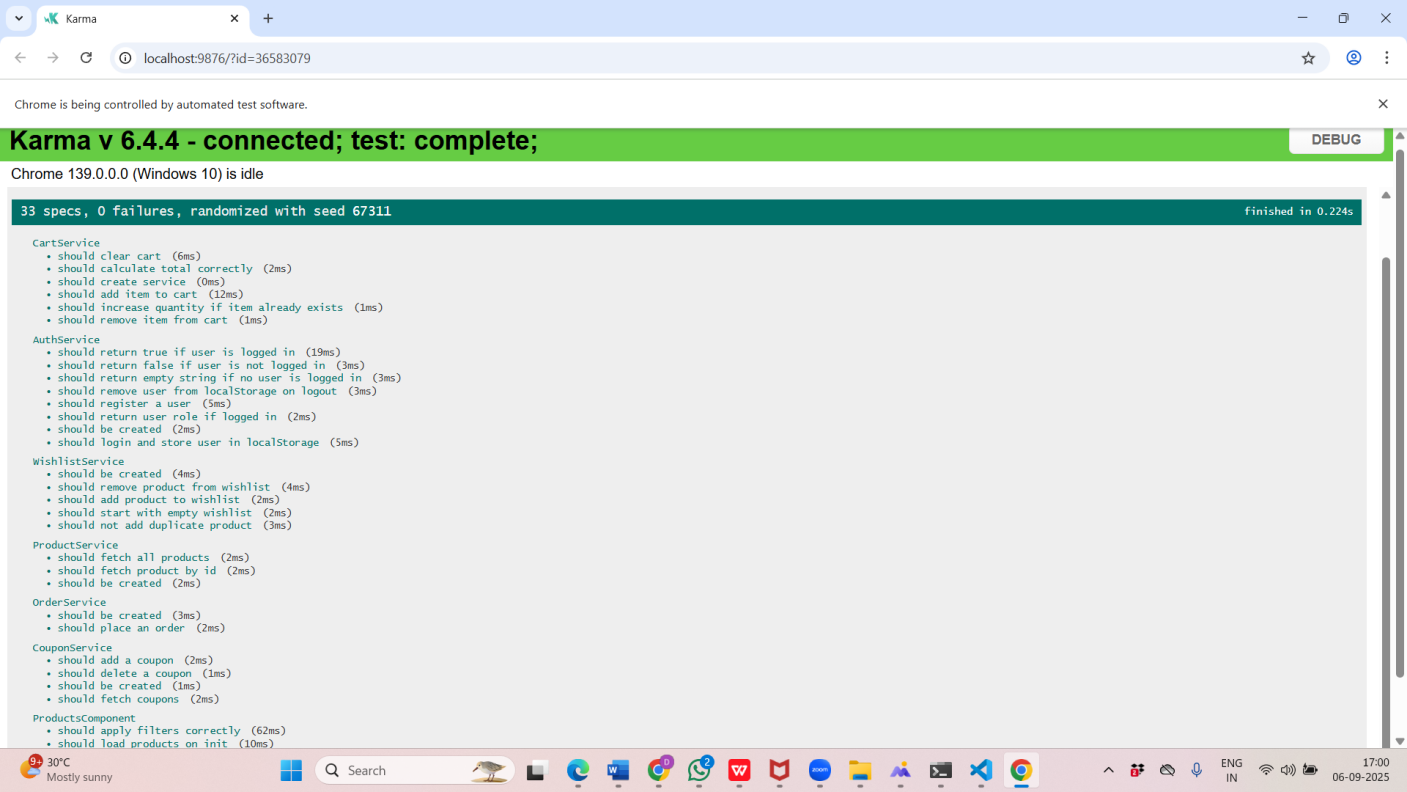
**FRONTEND CODE :**



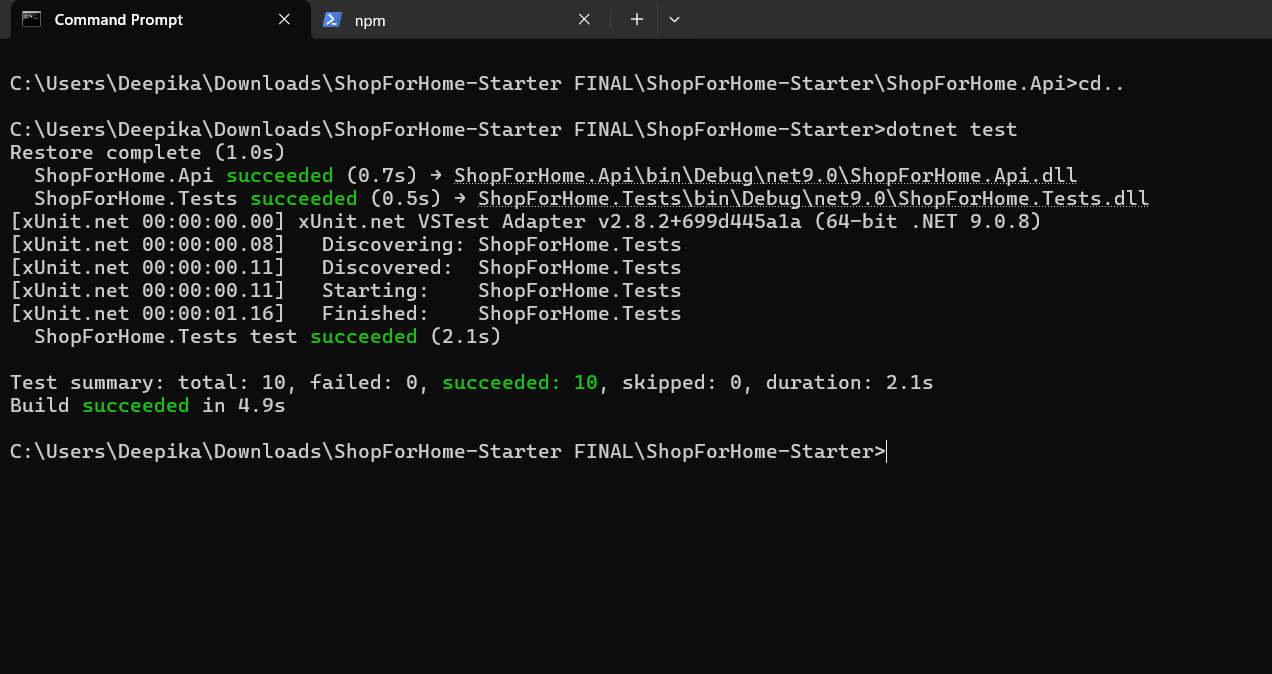
**BACKEND CODE :**



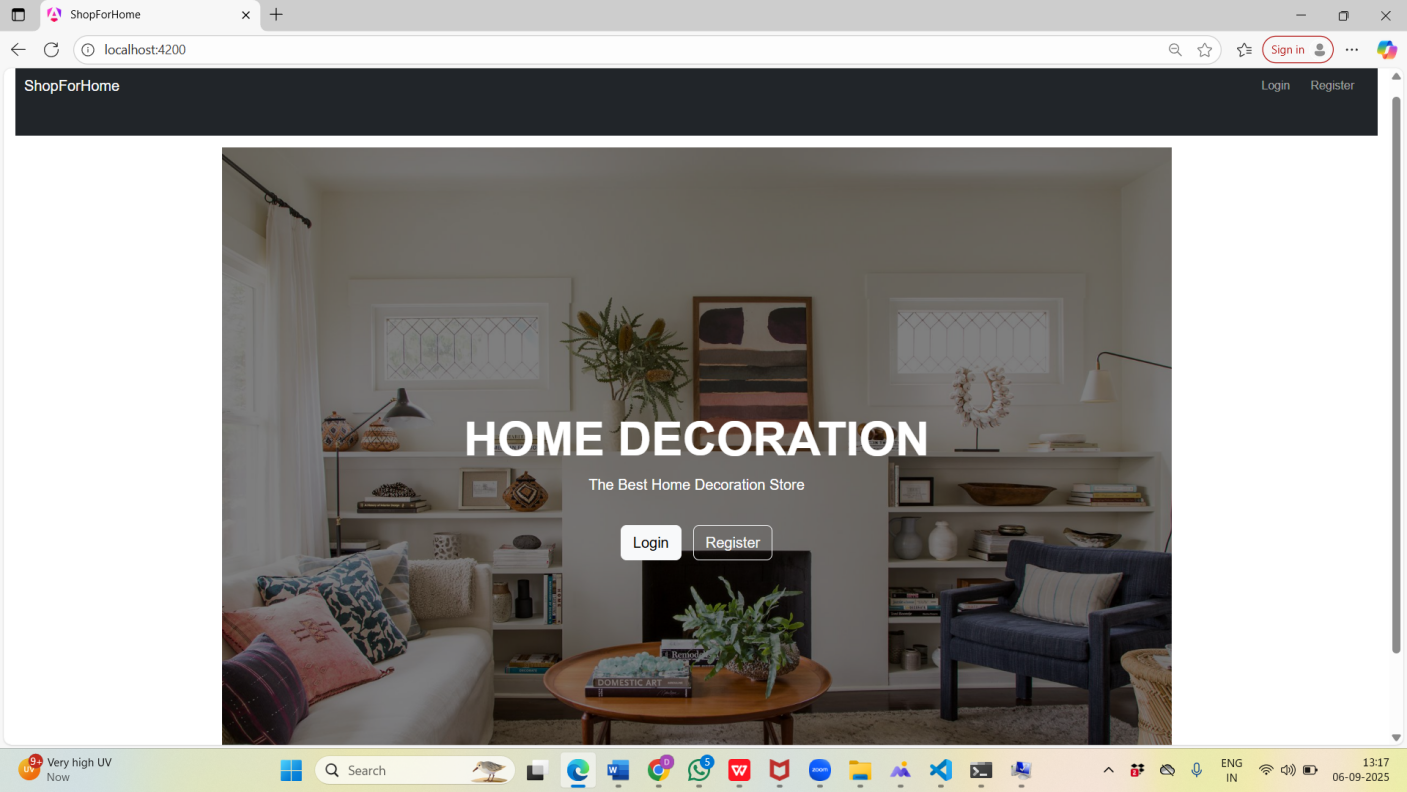
**FRONTEND TESTING :**



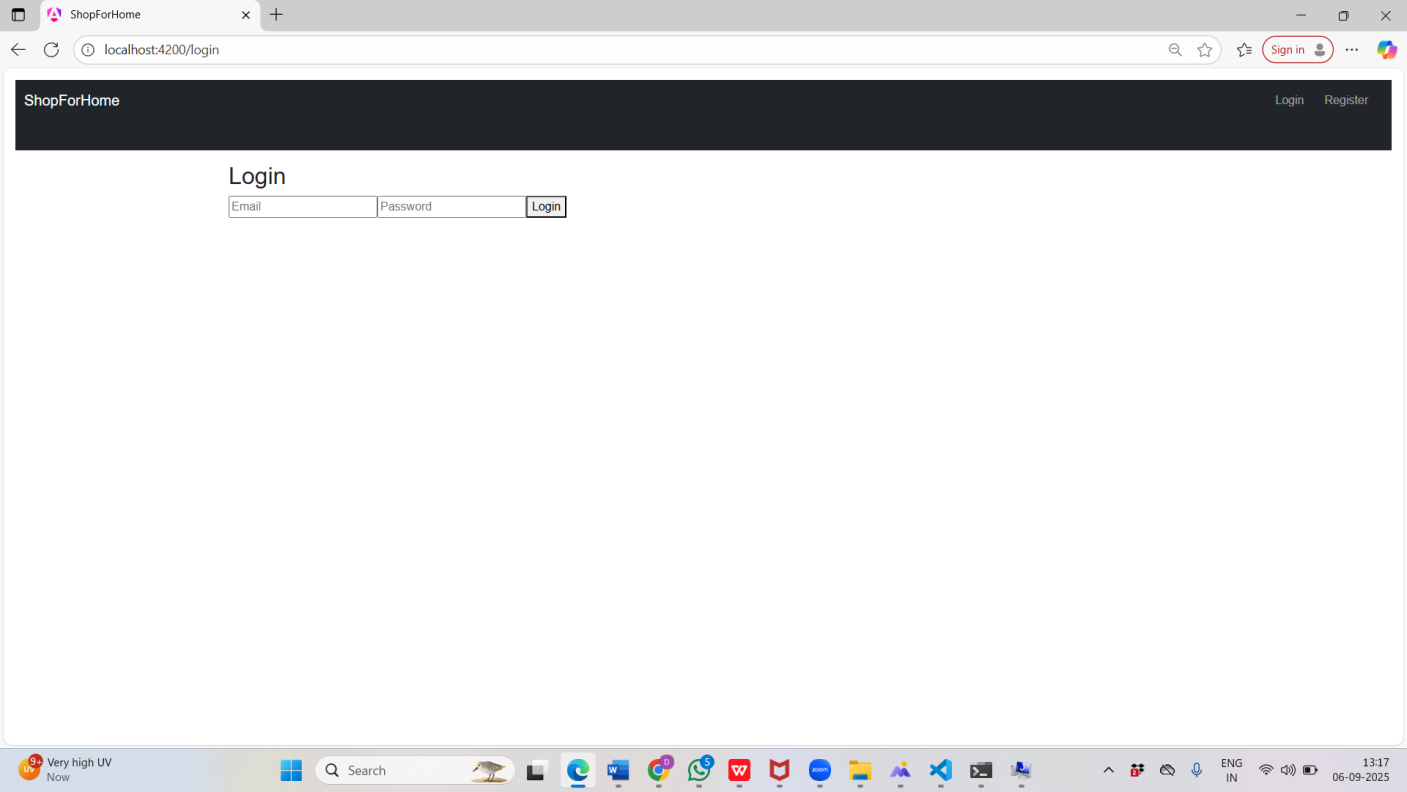
**BACKEND TESTING :**

****

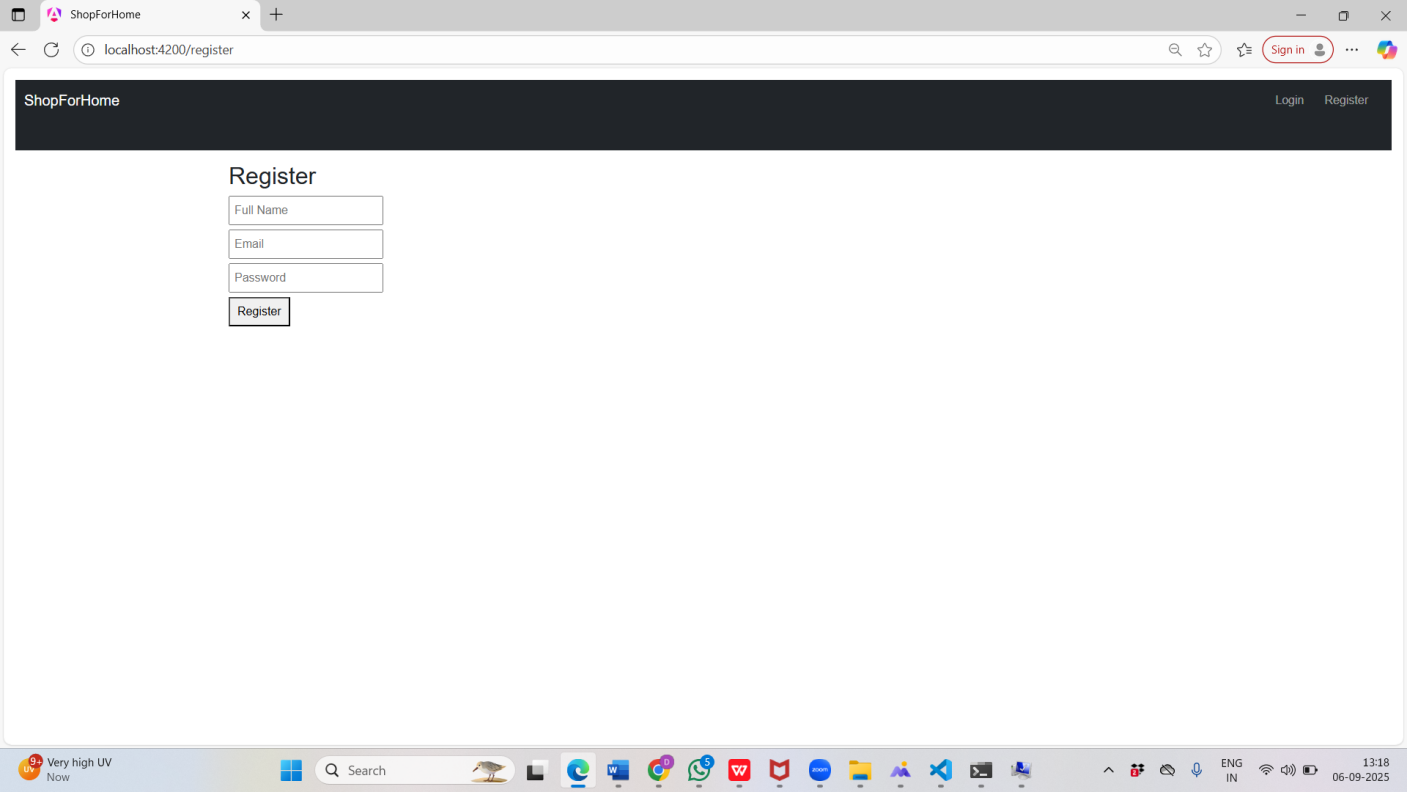
**HOME PAGE:**



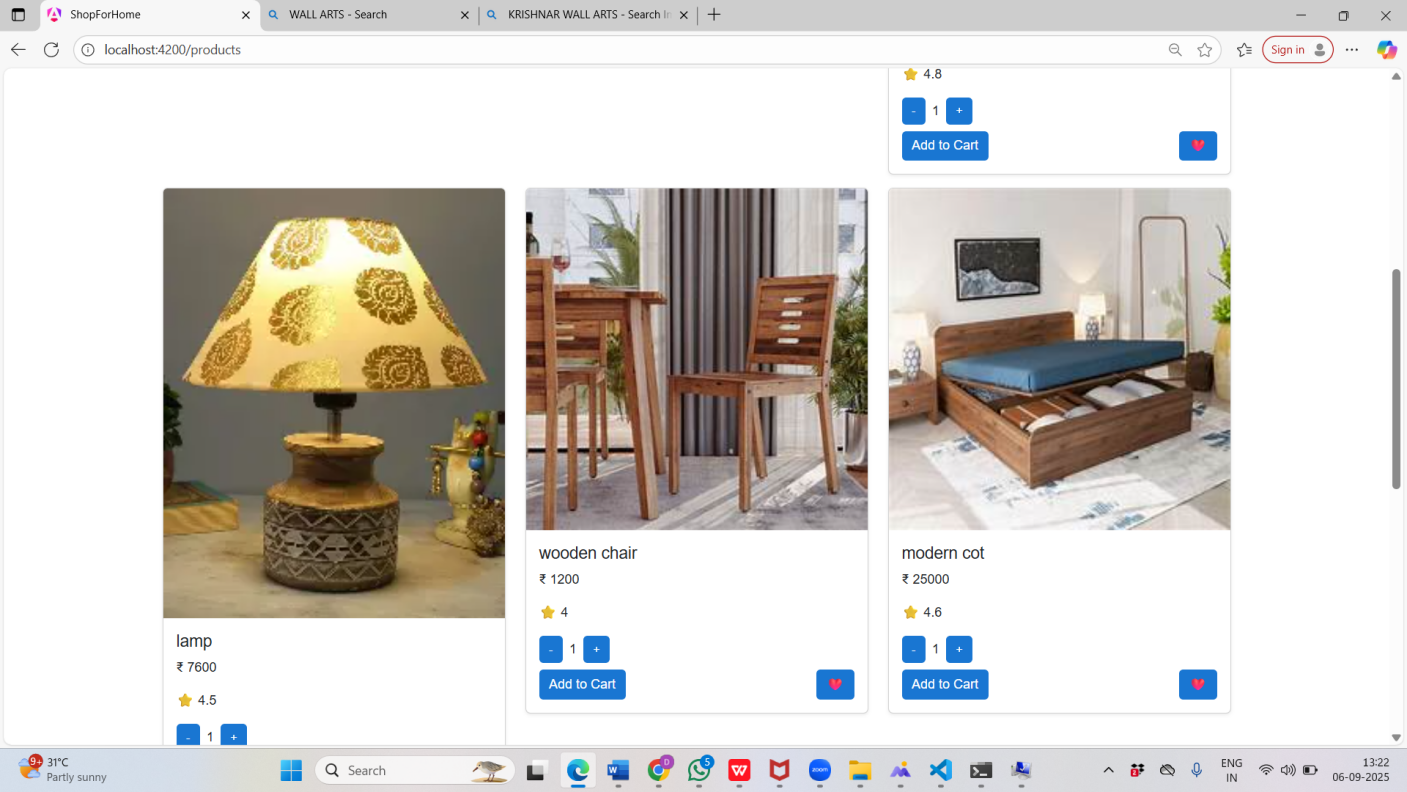
**LOGIN PAGE:**



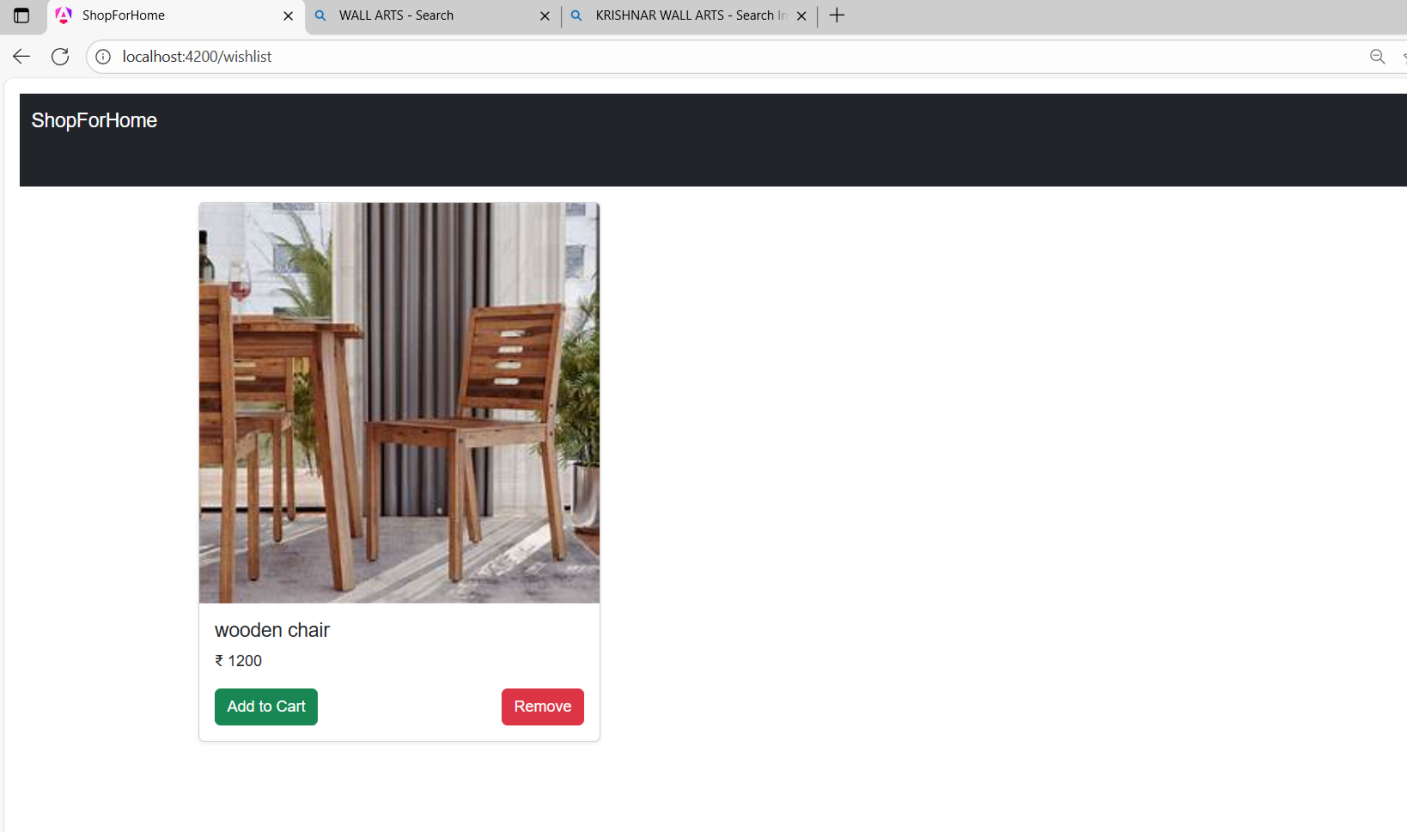
**REGISTER PAGE:**

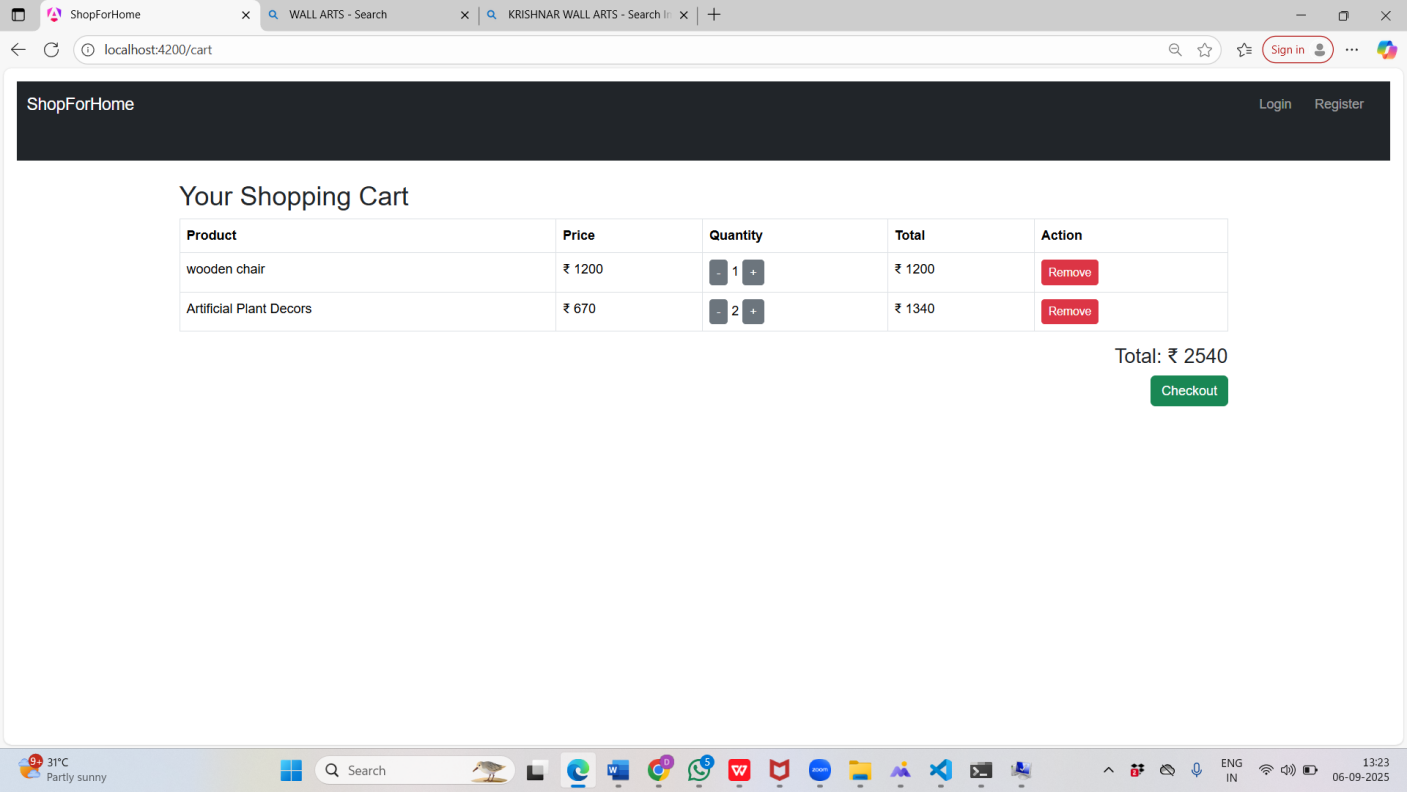


**USER PRODUCTS PAGE :**

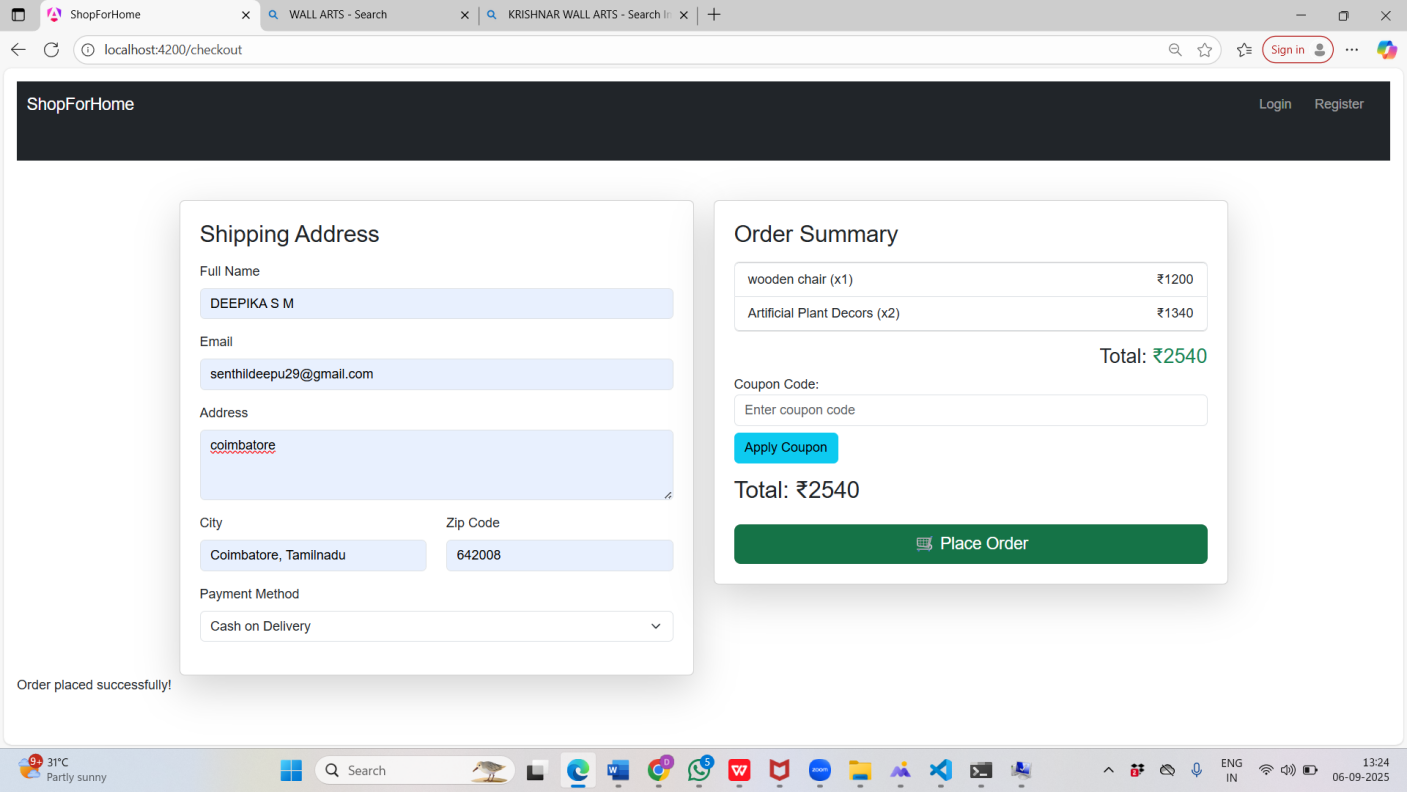


**USER WISHLIST PAGE :**

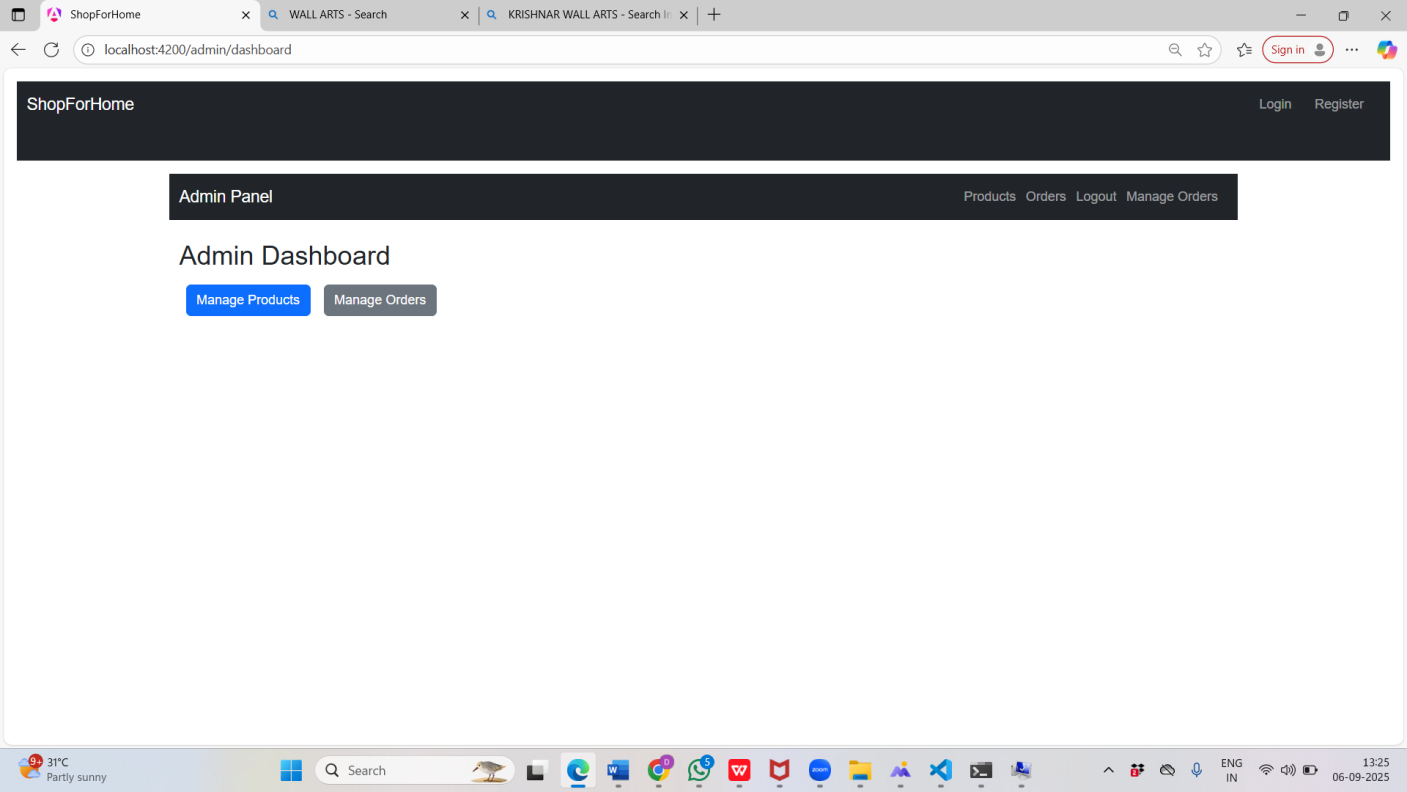
****

**CARTPAGE :**

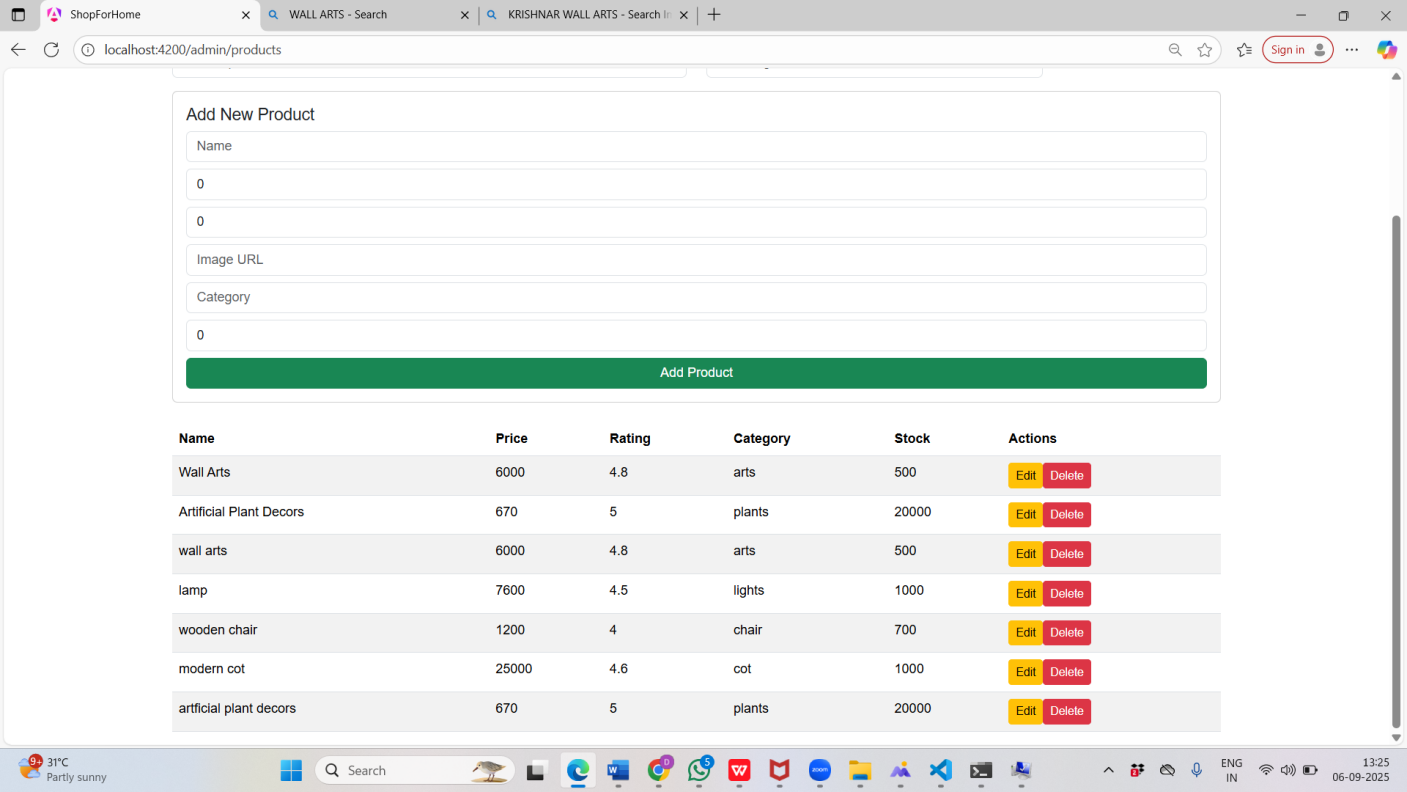
**CHECKOUT PAGE :**



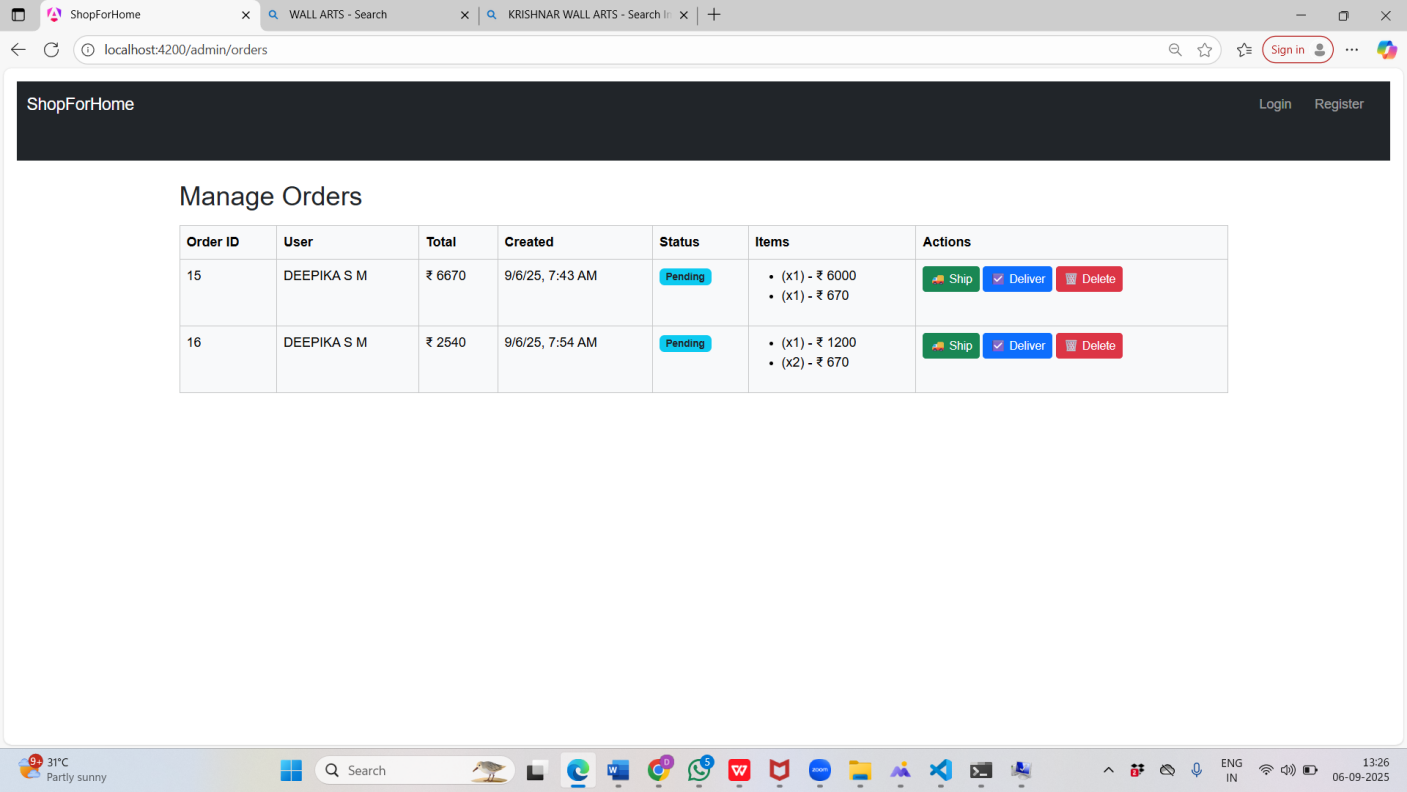
**ADMIN DASHBOARD:**



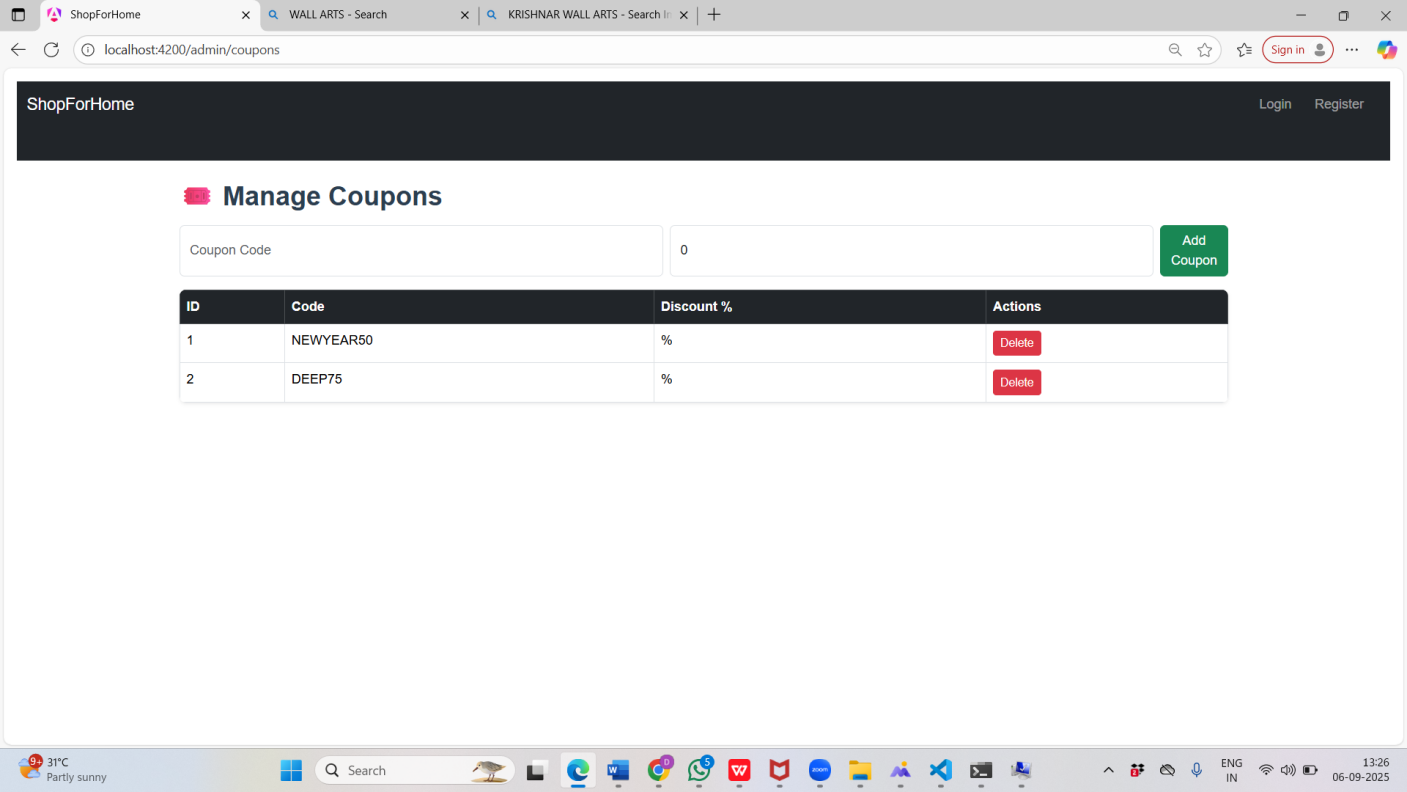
**ADMIN PRODUCTS:**



**ADMIN ORDERS:**



**ADMIN COUPONS:**



**CONCLUSION**

The *ShopForHome* project successfully demonstrates a complete e-commerce application with features such as authentication, product management, shopping cart, wishlist, coupons, and order placement. Using **ASP.NET Core Web API** for the backend and **Angular** for the frontend, the system ensures secure, responsive, and user-friendly operations. Testing with **Jasmine/Karma** and backend validations improved reliability and performance. Overall, the project not only meets its functional requirements but also provides practical experience in full-stack development, testing, and modern web application design.

Furthermore, this project highlights the importance of integrating different technologies into a seamless solution. By applying real-world scenarios such as managing users, securing transactions, and providing personalized shopping experiences, the system reflects the core aspects of modern e-commerce platforms. The knowledge gained from developing and testing this application lays a strong foundation for future projects and enhancements, such as adding payment gateways, advanced analytics, or cloud deployment for scalability.