**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**

**BÁO CÁO**

KỸ THUẬT LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

**ĐỀ TÀI: Website Bán đồ dùng công nghệ**

| Họ và tên: | Ngô Quang Minh |
| --- | --- |
| Mã sinh viên: | B19DCCN432 |
| Giảng viên: | Nguyễn Văn Tiến |

Hà Nội – 2023

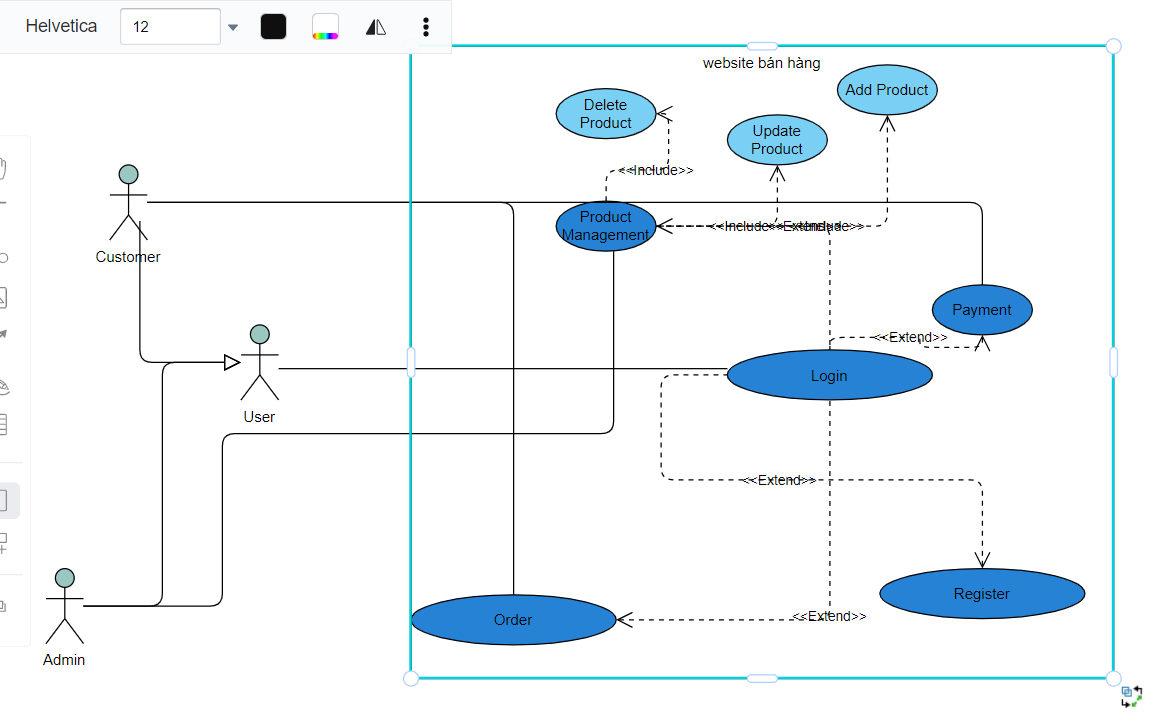
**Nội Dung Báo Cáo**

1. **Đặc Tả**

**1. Nghiệp Vụ**

* Website có 2 actor tham gia là Customer và Admin, cùng đăng nhập vào hệ thống và được gọi là User.
* User có những chức năng sau:
* xem danh sách các mặt hàng được bán trên trang web.
* xem chi tiết mặt hàng
* đăng nhập vào hệ thống
* Actor Customer có thể dùng các chức năng sau
* đăng ký tài khoản người dùng
* thêm vào giỏ hàng sản phẩm trên trang web
* đặt mua sản phẩm
* tiến hành thanh toán qua ví điện tử
* Actor Admin có thể dùng các chức năng sau:
* xem thông tin người dùng
* thêm sửa xóa mặt hàng
* xem thông tin các đơn hàng

**2. Use Case**



1. **BackEnd của hệ thống:**
2. **BE của hệ thống sử dụng FrameWork Spring trên IDE NetBeans**
3. **CSDL được sử dụng để lưu trữ là MySql**
4. **Spring - Spring Boot:**

* **giới thiệu: Spring Boot là một dự án phát triển bởi ngôn ngữ java trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.**

**Để phát triển một ứng dụng web cơ bản Hello World sử dụng Spring framework bạn sẽ cần ít nhất 5 công đoạn sau;**

* **Tạo một project sử dụng Maven với các dependency cần thiết của Spring MVC và Servlet API.**
* **Một tập tin web.xml để khai báo DispatcherServlet của Spring MVC.**
* **Một tập tin cấu hình của Spring MVC.**
* **Một class Controller trả về một trang “Hello World” khi có request đến.**
* **Cuối cùng là phải có một web server dùng để triển khai ứng dụng lên chạy.**

**Trong các công đoạn này, chỉ có công đoạn tạo một class Controller thì có thể khác cho các ứng dụng khác nhau vì mỗi ứng dụng có một yêu cầu khác nhau. Còn các công đoạn khác thì như nhau.**

**Giờ đây với Spring Boot, chúng ta có thể tạo dự án Spring một cách nhanh chóng và cấu hình cũng đơn giản dùng Sublime Text để phát triển luôn khỏi cần cài đặt eclipse hay netbean nặng bỏ bà.  
Dưới đây là một số tính năng nổi bật của Spring Boot:**

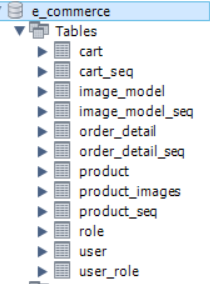
* **Tạo các ứng dụng Spring độc lập**
* **Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần phải deploy ra file WAR)**
* **Các starter dependency giúp việc cấu hình Maven đơn giản hơn**
* **Tự động cấu hình Spring khi cần thiết**
* **Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML …**
* **nội dung BE: những phần cần chú ý trong BE bao gồm:**

1. **các phụ thuộc Dependency được sử dụng, gồm có:**

* **razorpay: cổng thanh toán quốc tế, hỗ trợ những api thanh toán qua thẻ, visa,...**
* **spring jpa: sử dụng để khởi tạo bảng qua mã nguồn, kết nối và giao tiếp với csdl**
* **mysql-connector-j: cung cấp driver cần thiết để project kết nối với csdl**
* **lombok: một dependency cung cấp tác dụng rút gọn mã nguồn**
* **JJWT: cung cấp các công cụ cho việc tạo, xác minh và xử lý JSON Web Tokens (JWT) trong ứng dụng Java, bao gồm cả các ứng dụng Spring.**

1. **hệ thống các thực thể và các bảng trong csdl tương ứng cần quan tâm:**

* **các thực thể quan trọng chính: bao gồm User, Product, OrderDetail, TransactionDetail.**
* **những thực thể dẫn xuất: Cart, OrderInput, OrderProductQuanity.**
* **những thực thể hỗ trợ: Role, Image, JwtRequest/Response.**
* **các bảng trong csdl:**

****

**mối quan hệ giữa các bảng cần lưu ý:**

**Image và Product là 2 đối tượng riêng biệt trên BE, được đại diện bởi 2 bảng image\_model và product, kết nối với nhau bằng bảng product\_images lưu trữ id của những image thuộc product**

**role và user được đại diện bởi 2 bảng role và bảng user, trong đó bảng role sẽ tham chiếu đến user thông qua phép join để tạo ra bẳng user\_role. bảng user\_role chứa những user và role của họ.**

1. **JJWT cách vận hành của nó trong việc thực hiện Authen**

* lý thuyết: Spring hỗ trỡ handle các HTTP request với các servlet object

Và việc mong muốn cấu trúc project theo hương AOP của Spring FrameWork. Spring đưa ra kiến trúc Filter chain, nhằm truyền request qua các filter để sử lí các tác vụ ngoài nghiệp vụ chính. VD: logging, Authentication, Filter-data, …

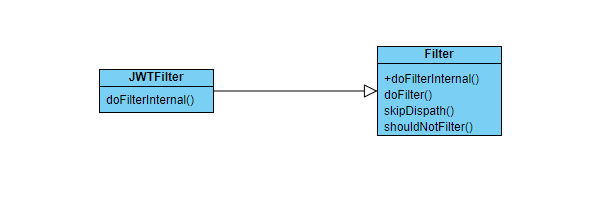
Web truyện hay sử dụng Design Pattern này nhằm xây dựng một model giúp authenticate các request



Như chúng ta thấy phía trên, request sẽ được sử lý qua các lớp filter để đến được tới lớp Servlet phía dưới. Việc này giúp chúng ta phân tách được các lớp khi sử lý request. Lúc này các lớp sử lý sẽ độc lập với nhau. Dễ dàng để kiểm thử và thay đổi logic hơn.

Việc thêm object filter chain sẽ được sử lý tự động qua việc scan các object bean của Spring.

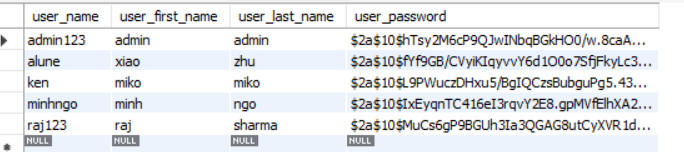
Trong tất các các use case thêm sửa xóa phía admin. Tất cả các request đều phải qua lớp filter của spring. Việc khởi tạo các object của spring



Việc insert object này vào chain filter sẽ do spring tự động quét.

* cách vận hành: JwtRequestFilter được WebSecurityConfiguration gọi đến, trong JwtRequestFilter có Autowired một Service gọi là JwtService. Service này có chức năng sinh Bearer Token từ Username và Pass, bằng cách tiêm AuthenticationManager và sử dụng phương thức .authenticate() của nó. từ Token này(có thời hạn là Session của User đó khi User còn đang ở trọng trạng thái login), hệ thống sẽ dùng nó để xác định role của user và cấp cho họ những role tương ứng.

1. phân chia role của các api, một số vấn đề về cors và Adapter của WebConfiguration:
2. encoder password bằng BCrypt: sử dụng hàm băm để mã hóa mật khẩu, với kết quả thu được trên db



1. **FrontEnd của hệ thống: Angular**
2. **giới thiệu: Angular là gì? Angular là một JavaScript framework dùng để viết giao diện web (Front-end), được phát triển bởi Google. Angular giúp lập trình viên xây dựng các ứng dụng trang đơn (single-page application) bằng cách sử dụng HTML và TypeScript một cách nhanh hơn.**
3. **một số Interceptor cần quan tâm:**

* **import {**
* **HttpErrorResponse,**
* **HttpEvent,**
* **HttpHandler,**
* **HttpInterceptor,**
* **HttpRequest,**
* **} from '@angular/common/http';**
* **import { Router } from '@angular/router';**
* **import { catchError } from 'rxjs/operators';**
* **import { Observable, throwError } from 'rxjs';**
* **import { UserAuthService } from '../\_services/user-auth.service';**
* **import { Injectable } from '@angular/core';**
* **@Injectable()**
* **export class AuthInterceptor implements HttpInterceptor {**
* **constructor(private userAuthService: UserAuthService,**
* **private router:Router) {}**
* **intercept(**
* **req: HttpRequest<any>,**
* **next: HttpHandler**
* **): Observable<HttpEvent<any>> {**
* **if (req.headers.get('No-Auth') === 'True') {**
* **return next.handle(req.clone());**
* **}**
* **const token = this.userAuthService.getToken();**
* **if(token) {**
* **req = this.addToken(req, token);**
* **}**
* **return next.handle(req).pipe(**
* **catchError(**
* **(err:HttpErrorResponse) => {**
* **console.log(err.status);**
* **if(err.status === 401) {**
* **this.router.navigate(['/login']);**
* **} else if(err.status === 403) {**
* **this.router.navigate(['/forbidden']);**
* **}**
* **return throwError("Some thing is wrong");**
* **}**
* **)**
* **);**
* **}**

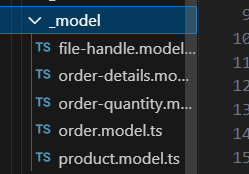
* **private addToken(request:HttpRequest<any>, token:string) {**
* **return request.clone(**
* **{**
* **setHeaders: {**
* **Authorization : `Bearer ${token}`**
* **}**
* **}**
* **);**
* **}**
* **}**

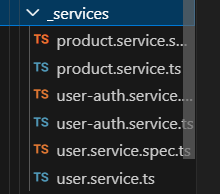
**trong đó, intercept() là phương thức chính thực hiện các thay đổi cần thiết. Nó kiểm tra xem request có đang yêu cầu token không và thêm token nếu có.**

**Khi có lỗi, nó sử dụng catchError từ rxjs/operators để xử lý lỗi. Nó kiểm tra status code của lỗi và dựa vào đó, điều hướng người dùng đến trang login hoặc trang cấm truy cập.**

**addToken() là một phương thức riêng để thêm token vào header của request. Nó sử dụng request.clone() để tạo một bản sao mới của request và thêm token vào đó.**

* **các model trong module model của project:**

****

****

**vì đây là những Interceptor được tái sử dụng liên tục trong các module.**

* **xử lý trong phần login:**

**import { Component, OnInit } from '@angular/core';**

**import { NgForm } from '@angular/forms';**

**import { Router } from '@angular/router';**

**import { UserAuthService } from '../\_services/user-auth.service';**

**import { UserService } from '../\_services/user.service';**

**@Component({**

**selector: 'app-login',**

**templateUrl: './login.component.html',**

**styleUrls: ['./login.component.css'],**

**})**

**export class LoginComponent implements OnInit {**

**constructor(**

**private userService: UserService,**

**private userAuthService: UserAuthService,**

**private router: Router**

**) {}**

**ngOnInit(): void {}**

**login(loginForm: NgForm) {**

**this.userService.login(loginForm.value).subscribe(**

**(response: any) => {**

**this.userAuthService.setRoles(response.user.role);**

**this.userAuthService.setToken(response.jwtToken);**

**const role = response.user.role[0].roleName;**

**if (role === 'Admin') {**

**this.router.navigate(['/admin']);**

**} else {**

**this.router.navigate(['/user']);**

**}**

**},**

**(error) => {**

**console.log(error);**

**}**

**);**

**}**

**registerUser() {**

**this.router.navigate(['/register']);**

**}**

**}**

**ngOnInit(): Đây là một lifecycle hook, được gọi khi component được khởi tạo.( ở bước khởi tạo module Header)**

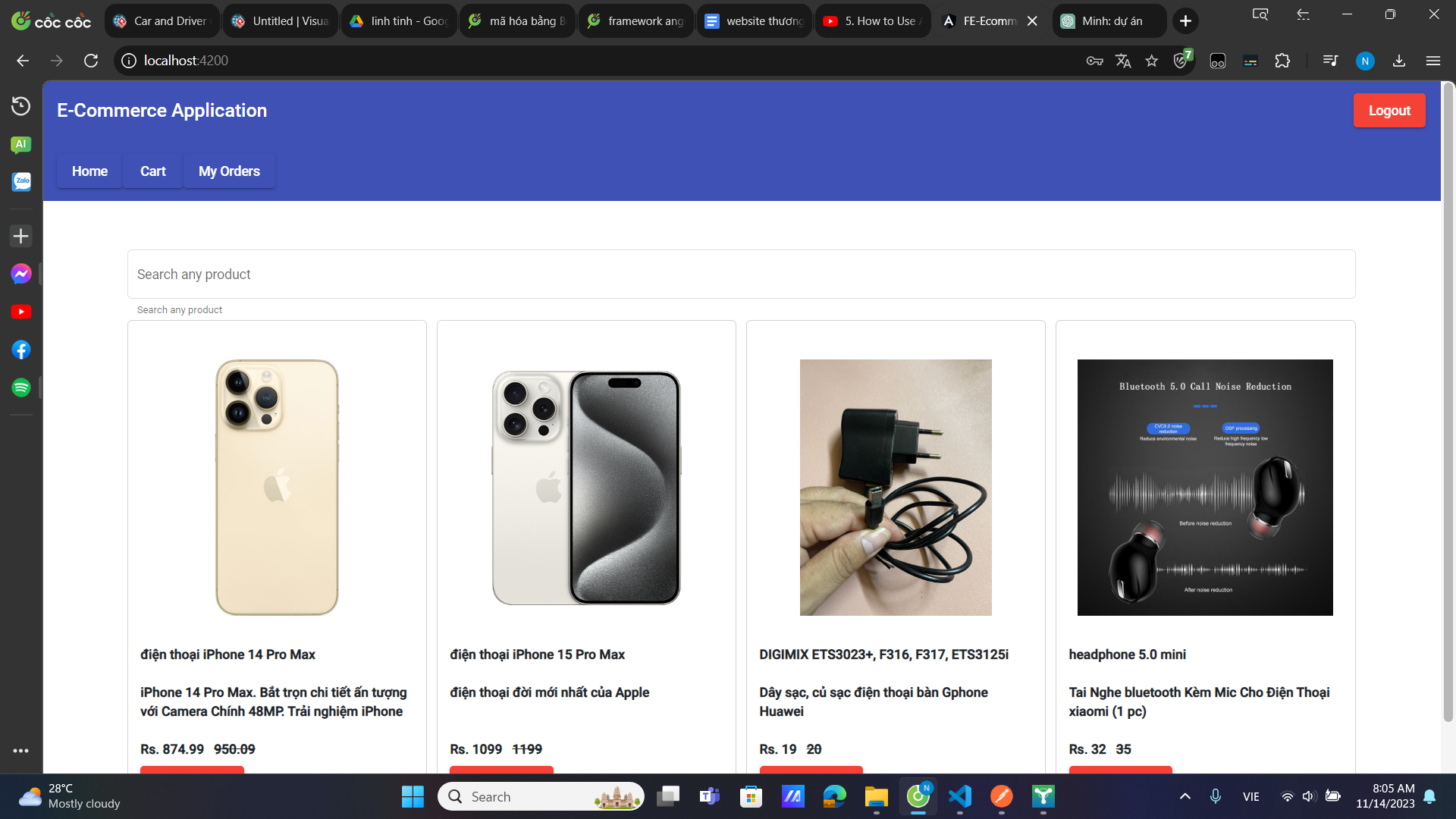
**login(loginForm: NgForm): Đây là hàm xử lý việc đăng nhập. Khi người dùng gửi form đăng nhập, hàm này được kích hoạt. Nó sử dụng UserService (được inject qua constructor) để thực hiện yêu cầu đăng nhập thông qua loginForm - thông tin đăng nhập người dùng nhập vào form.**

**Nếu đăng nhập thành công, phản hồi (response) từ server sẽ chứa thông tin người dùng và token JWT (JSON Web Token). UserAuthService được sử dụng để lưu trữ thông tin này và điều hướng người dùng đến trang '/admin' nếu họ là Admin, hoặc trang '/user' nếu không phải.**

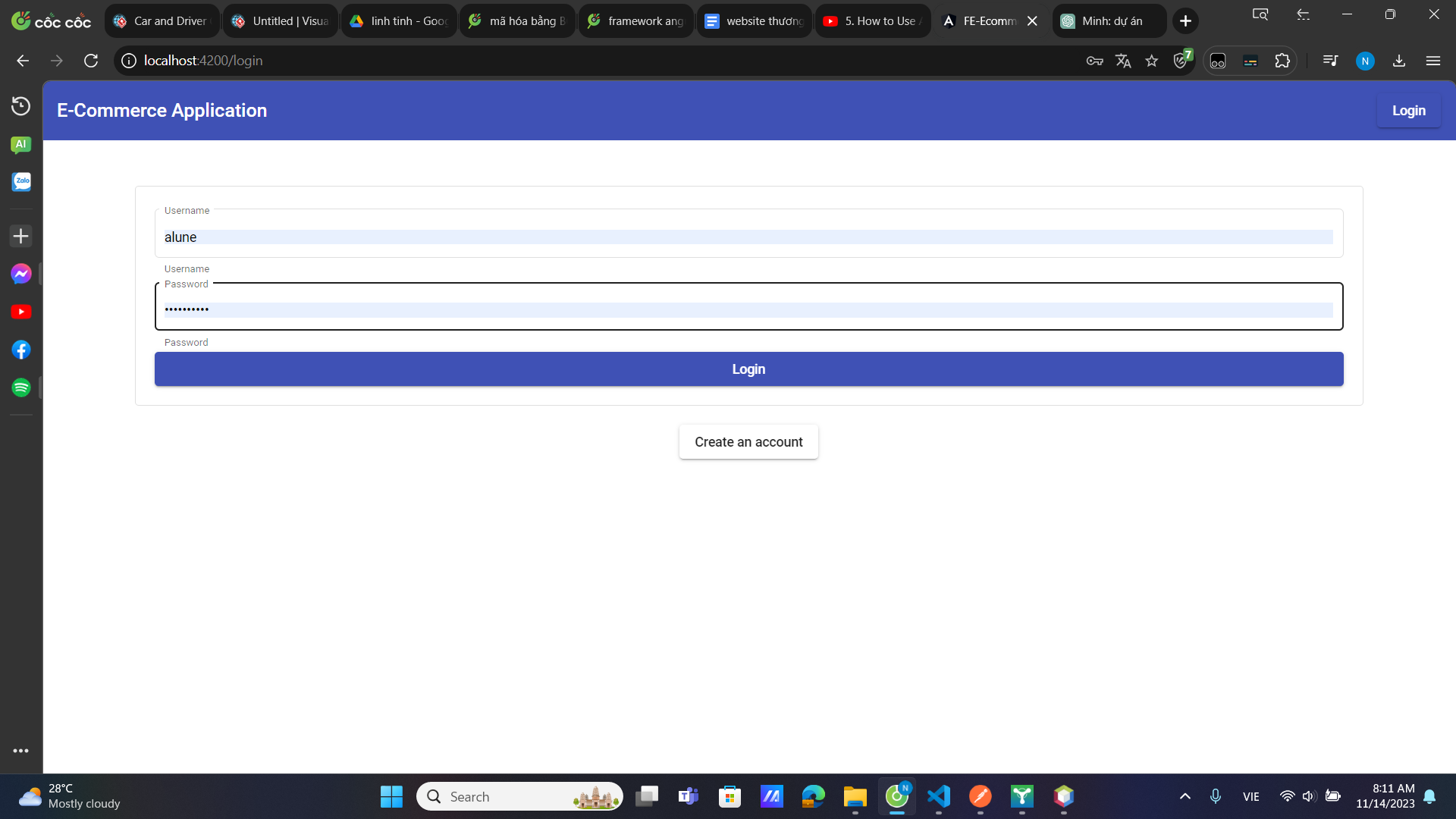
**Nếu xảy ra lỗi trong quá trình đăng nhập, lỗi sẽ được in ra console.**

**registerUser(): Hàm này đơn giản là điều hướng người dùng đến trang đăng ký '/register' khi họ muốn đăng ký tài khoản mới.**

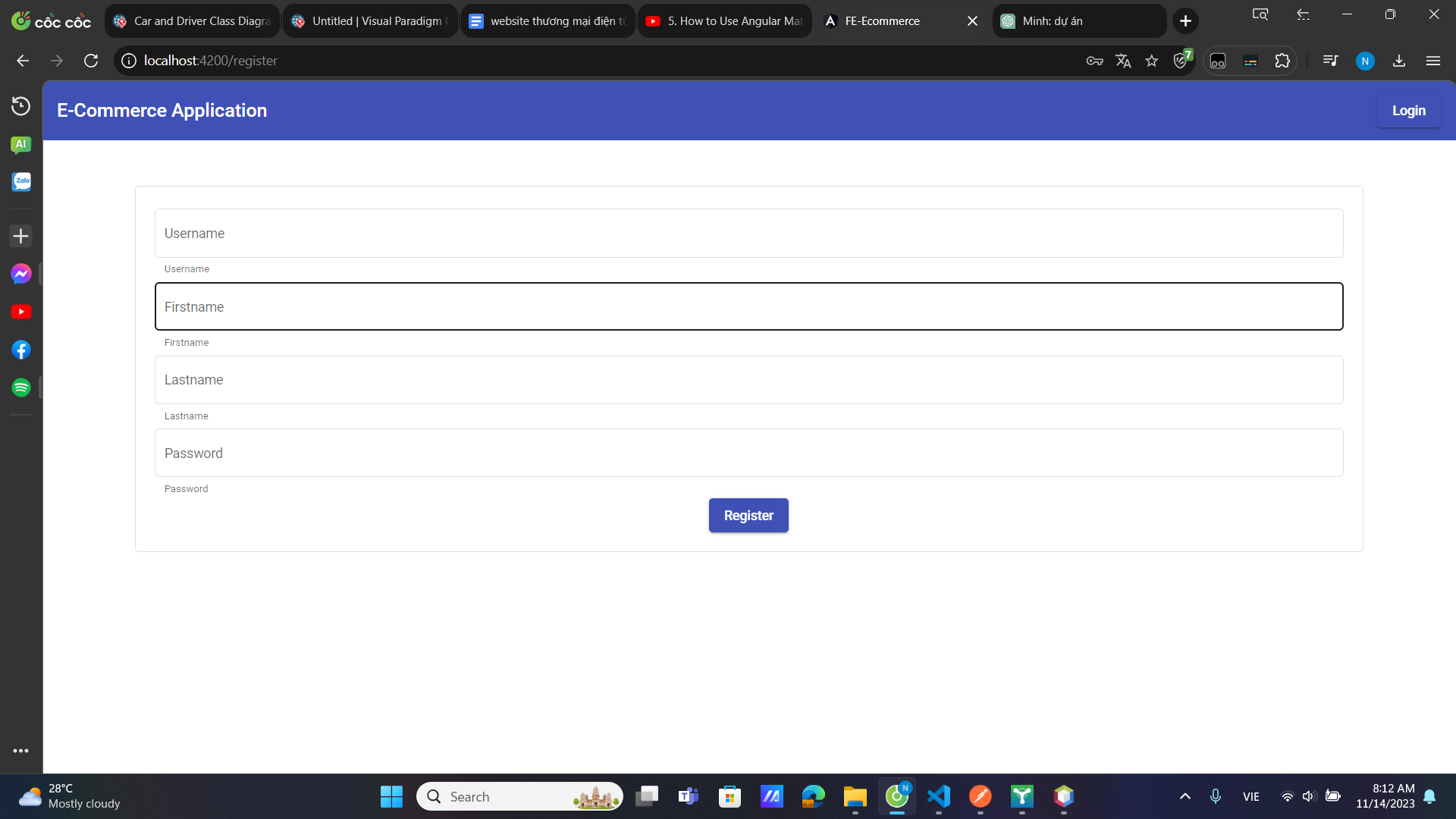
1. **giao diện trang web:**
2. **giao diện home:**

****

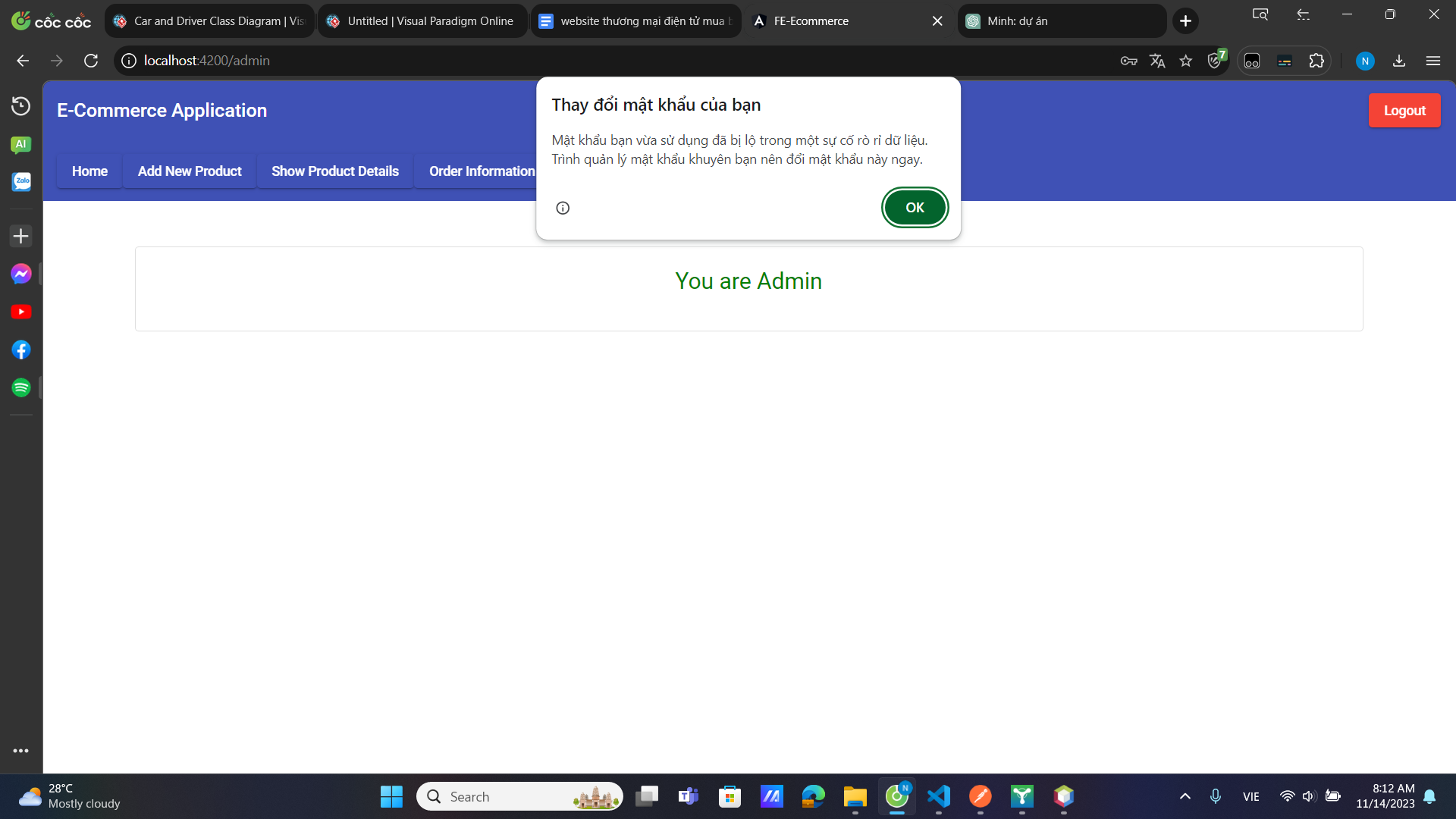
**giao diện đăng nhập:**

****

**giao diện đăng kí:**

****

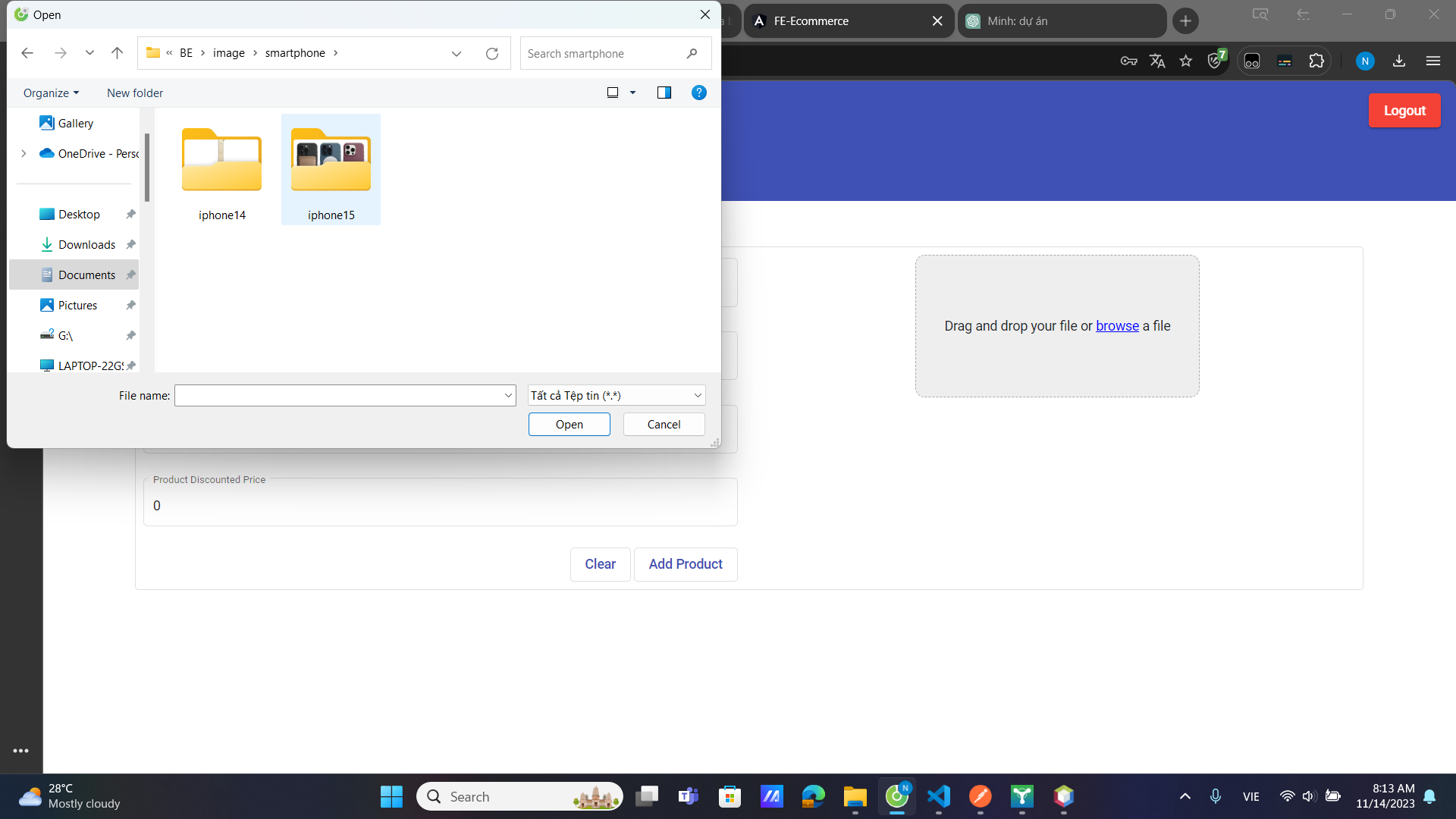
**đăng nhập với quyền admin:**

****

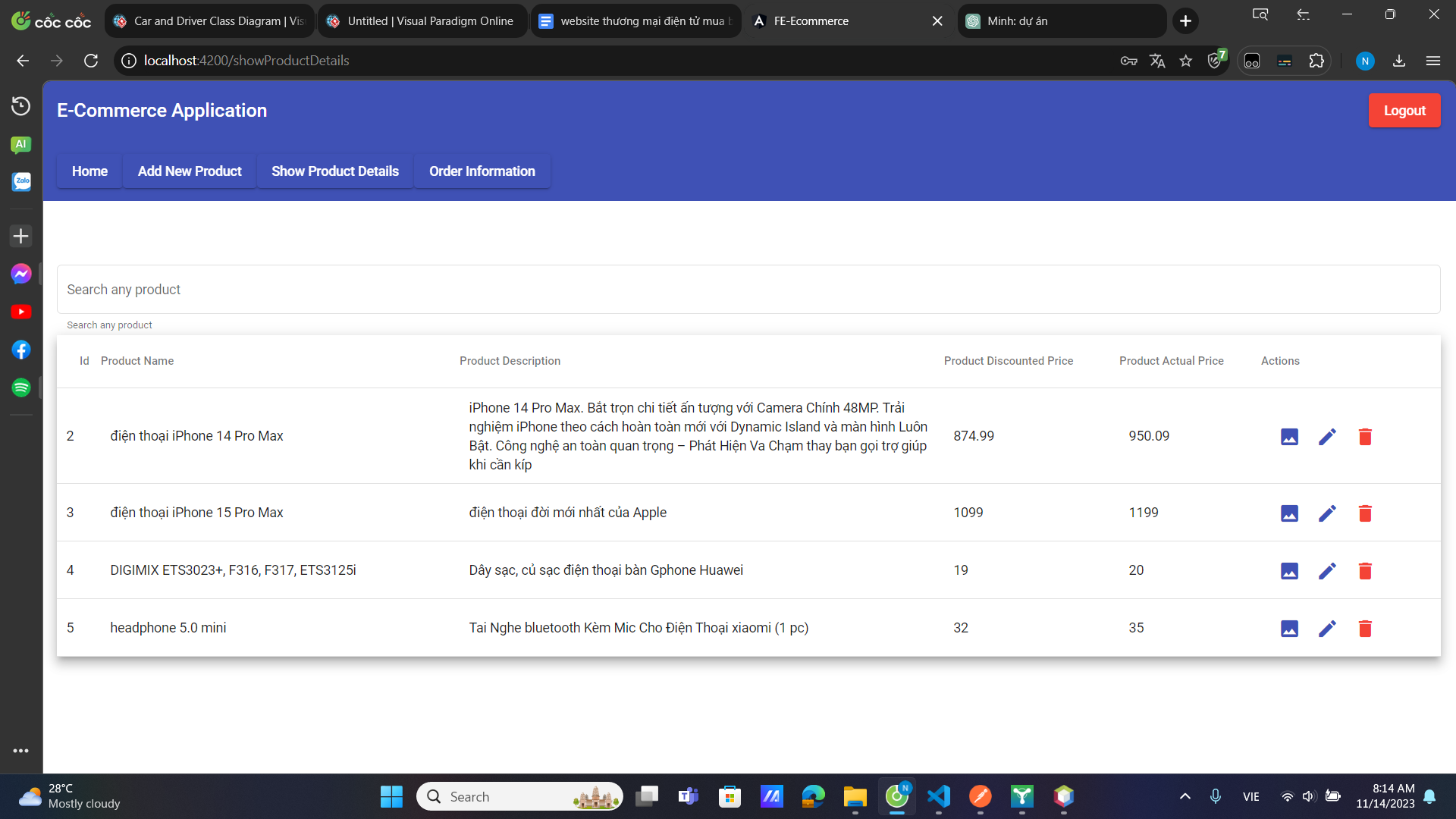
**giao diện thêm sp:**

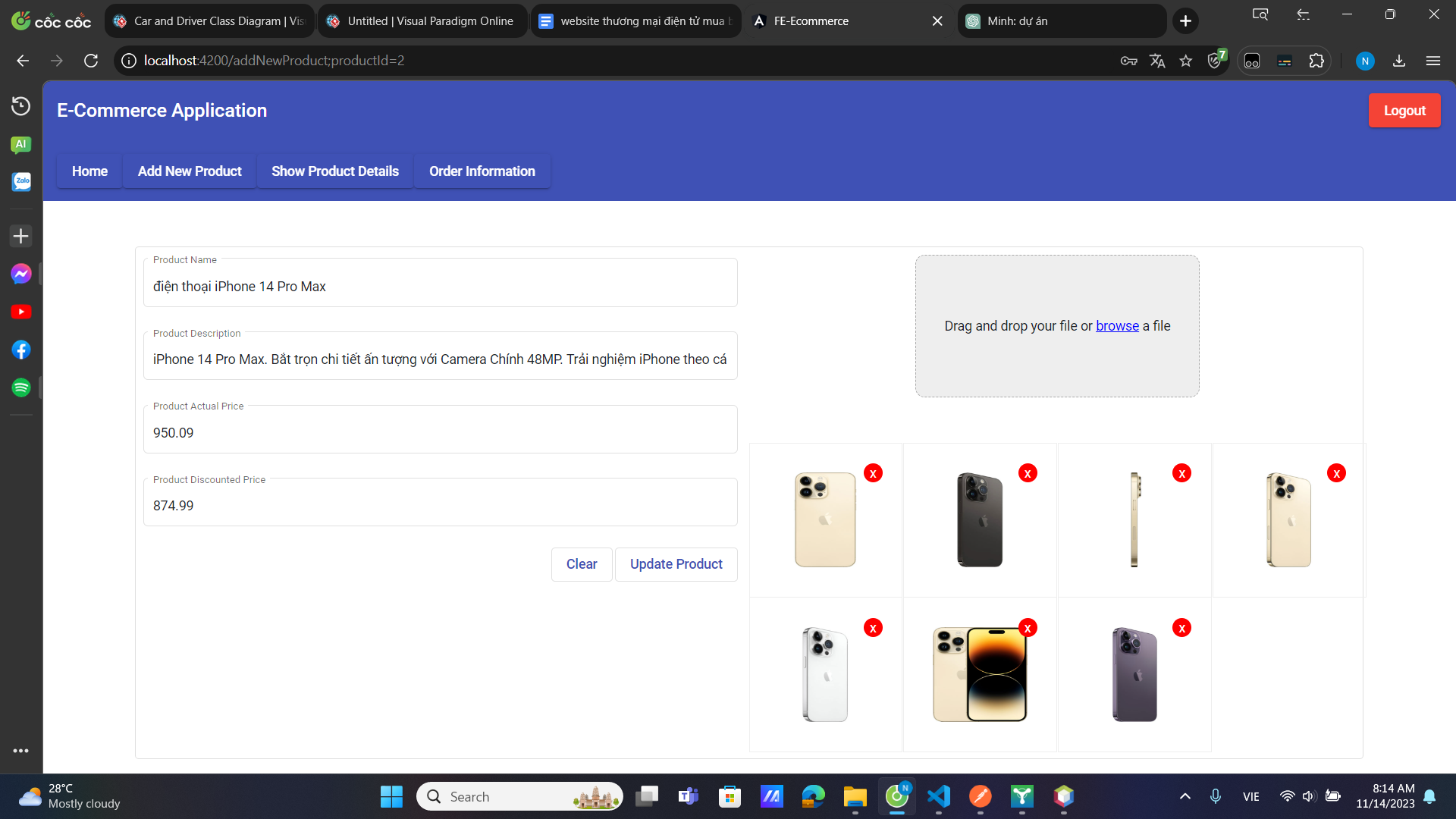
****

**chức năng thêm file ảnh của FE:**

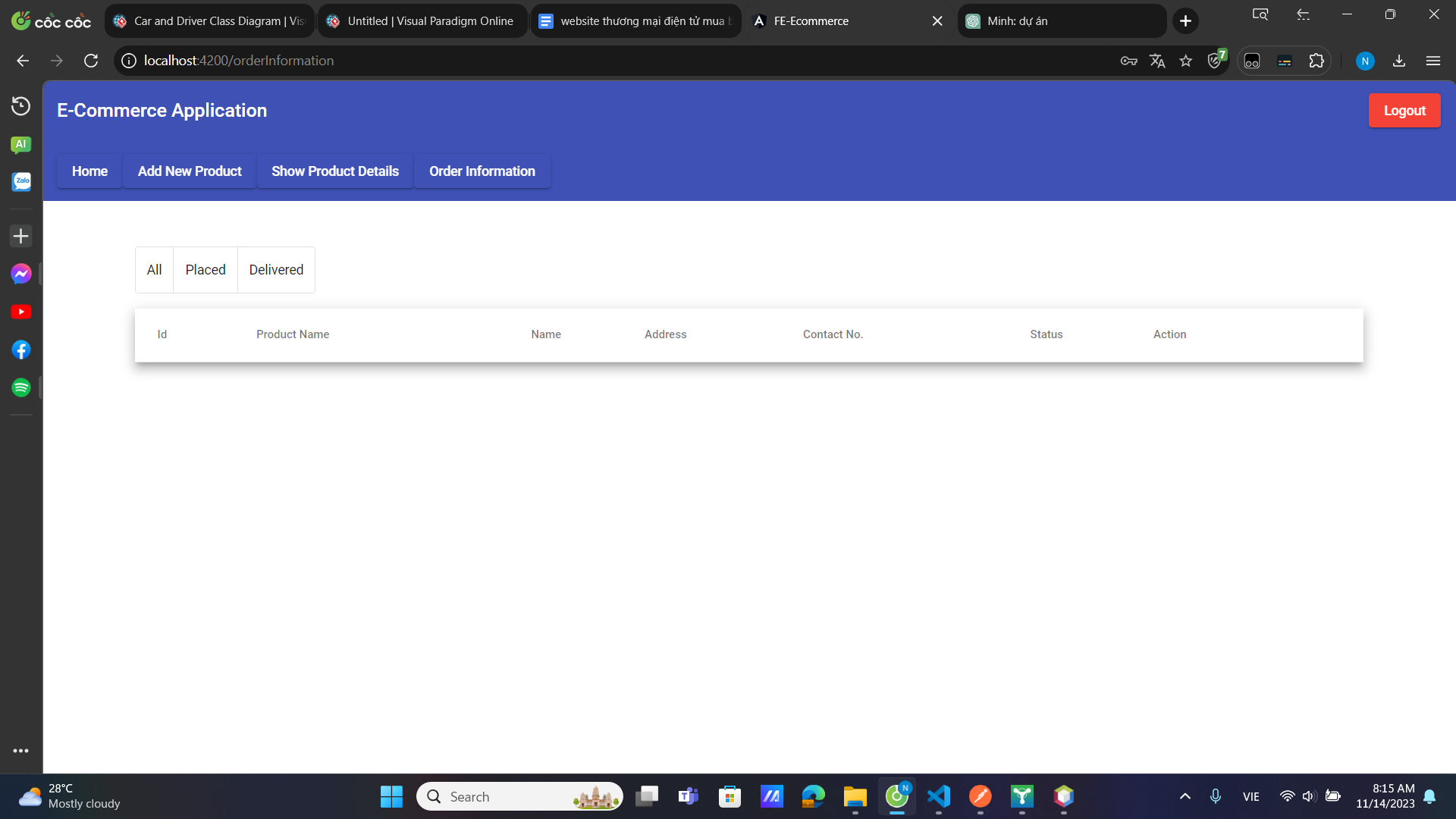
****

**xem chi tiết và sửa, xóa sản phẩm:**

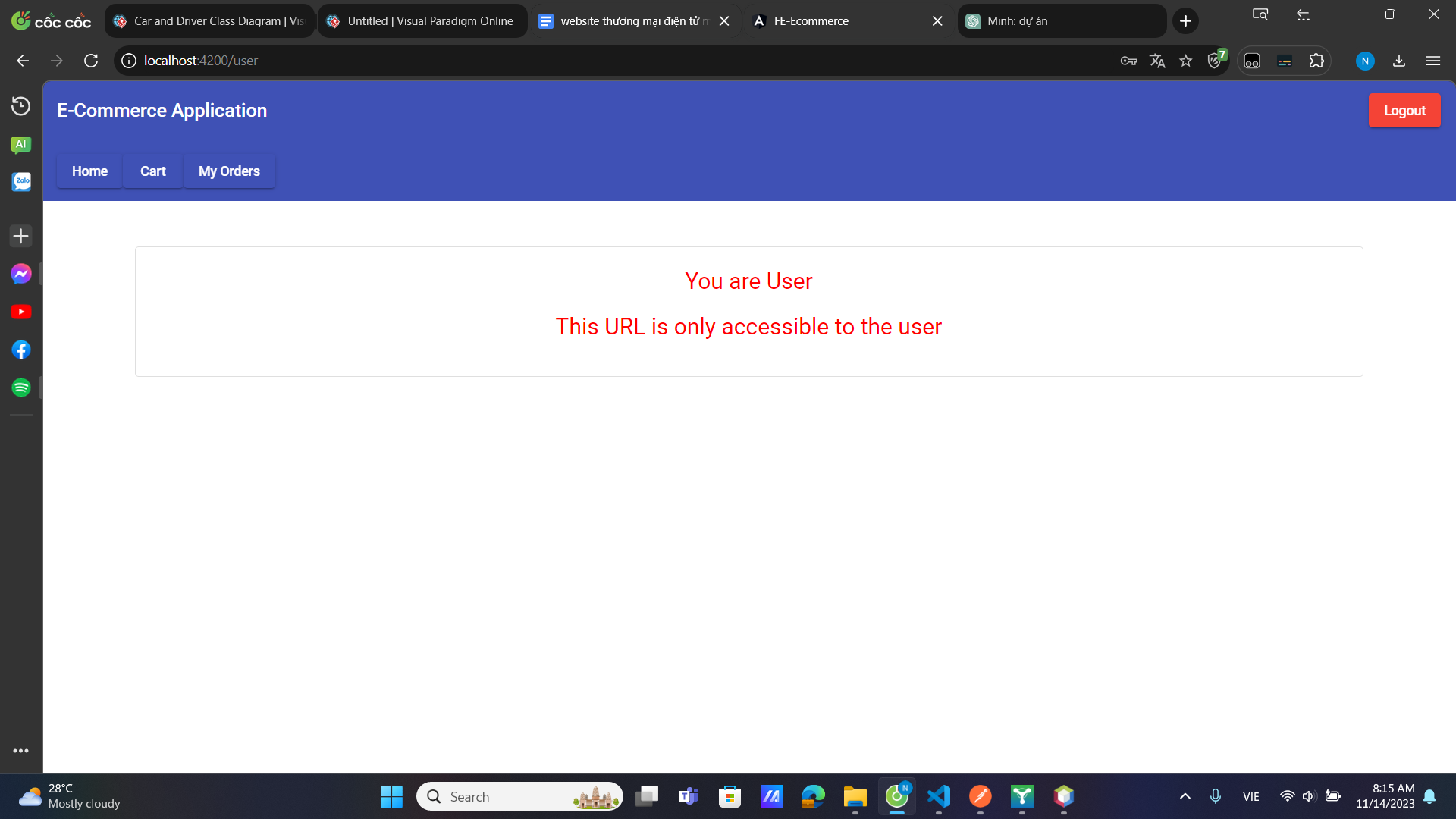
****

****

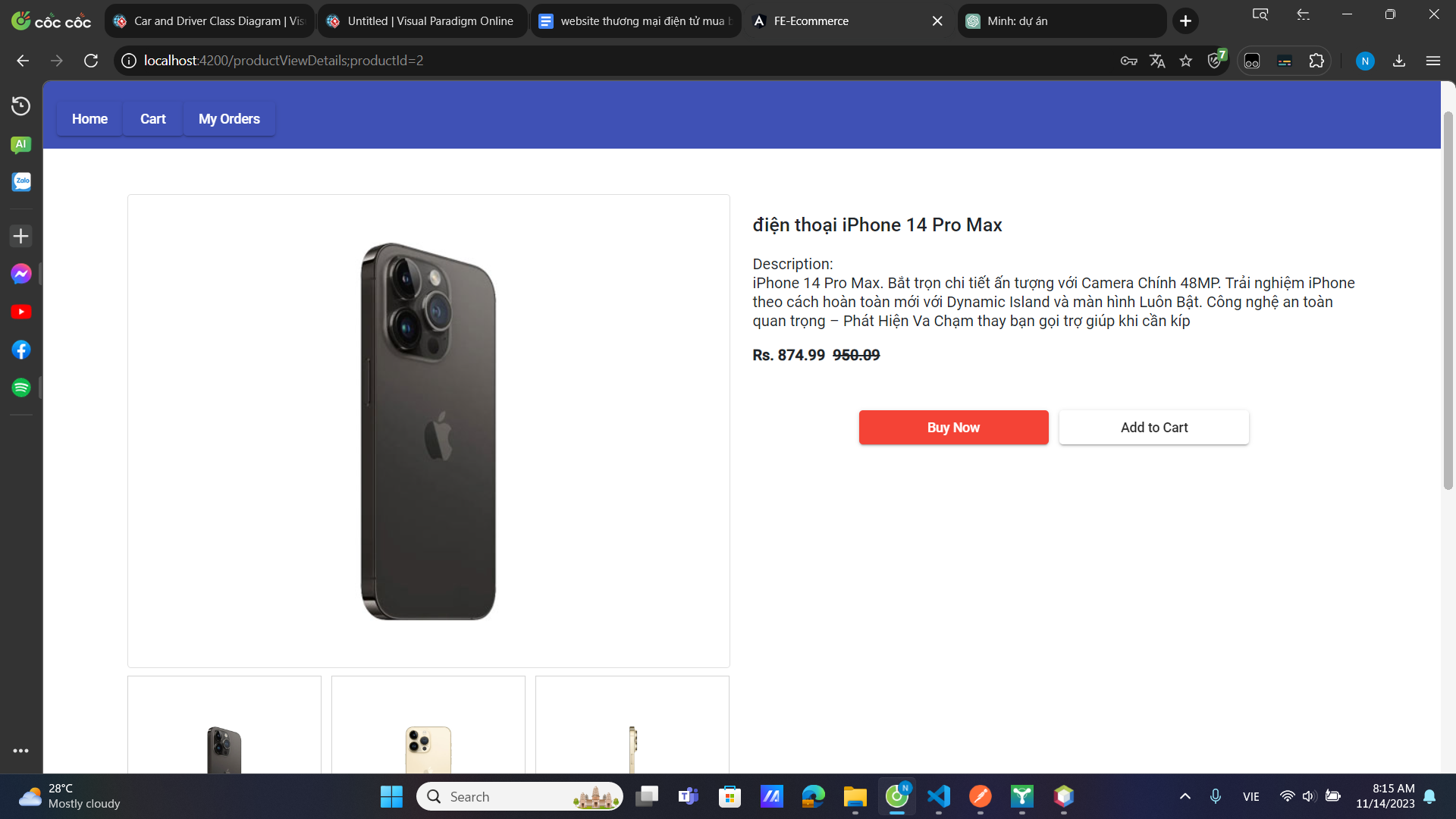
**xem giao dịch đã thực hiện:**

****

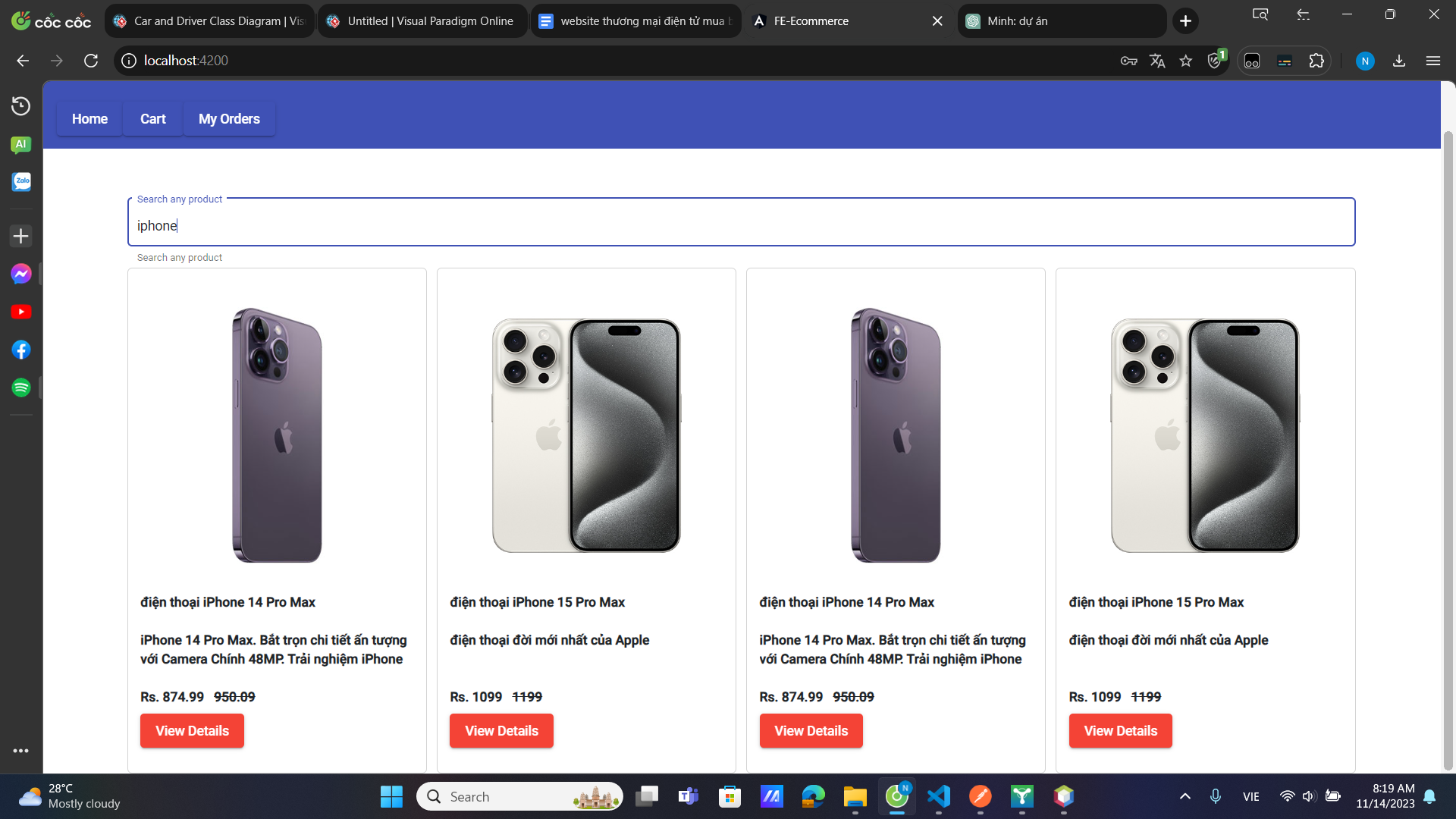
**giao diện người dùng:**

****

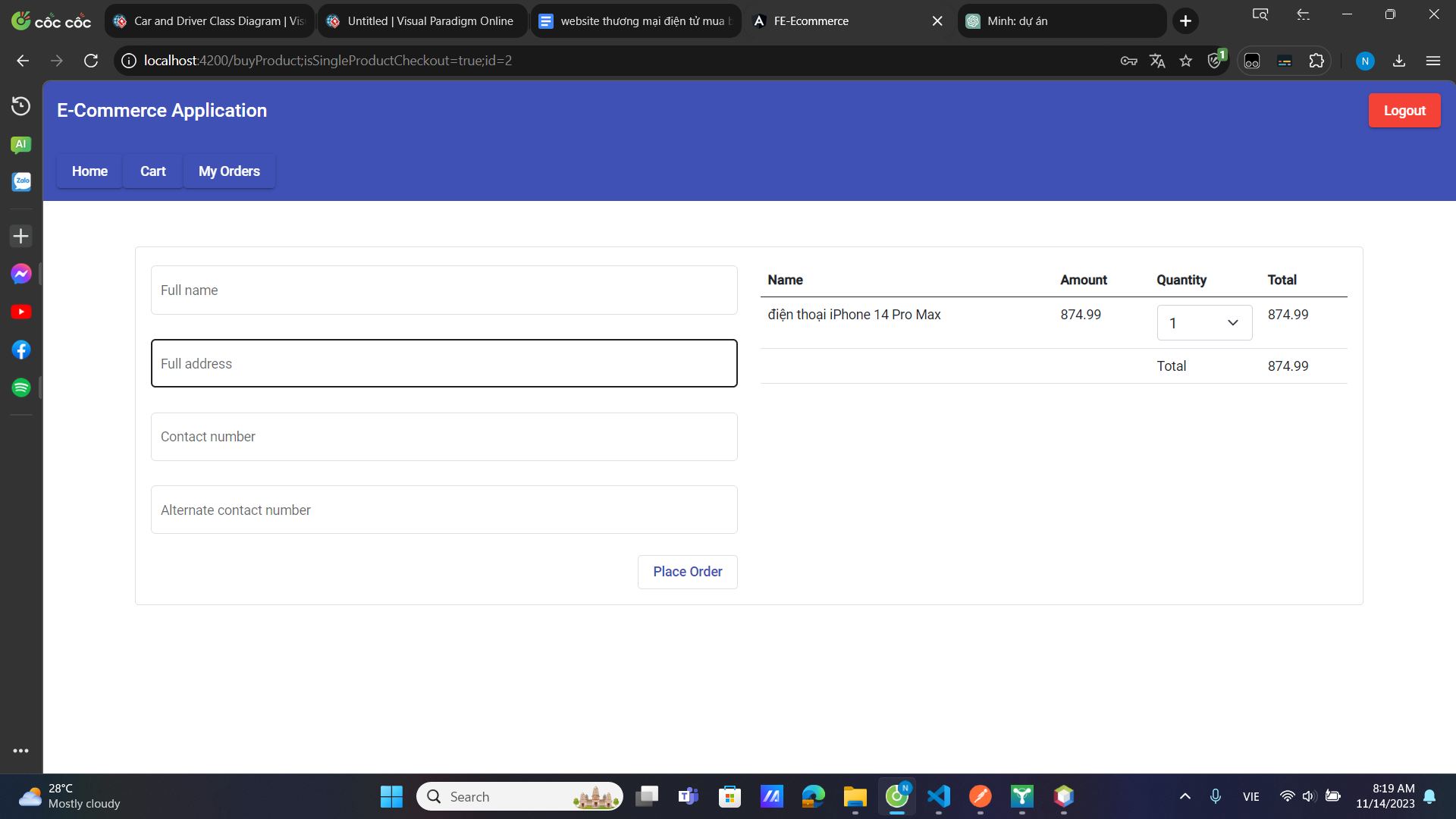
**xem chi tiết sp:**

****

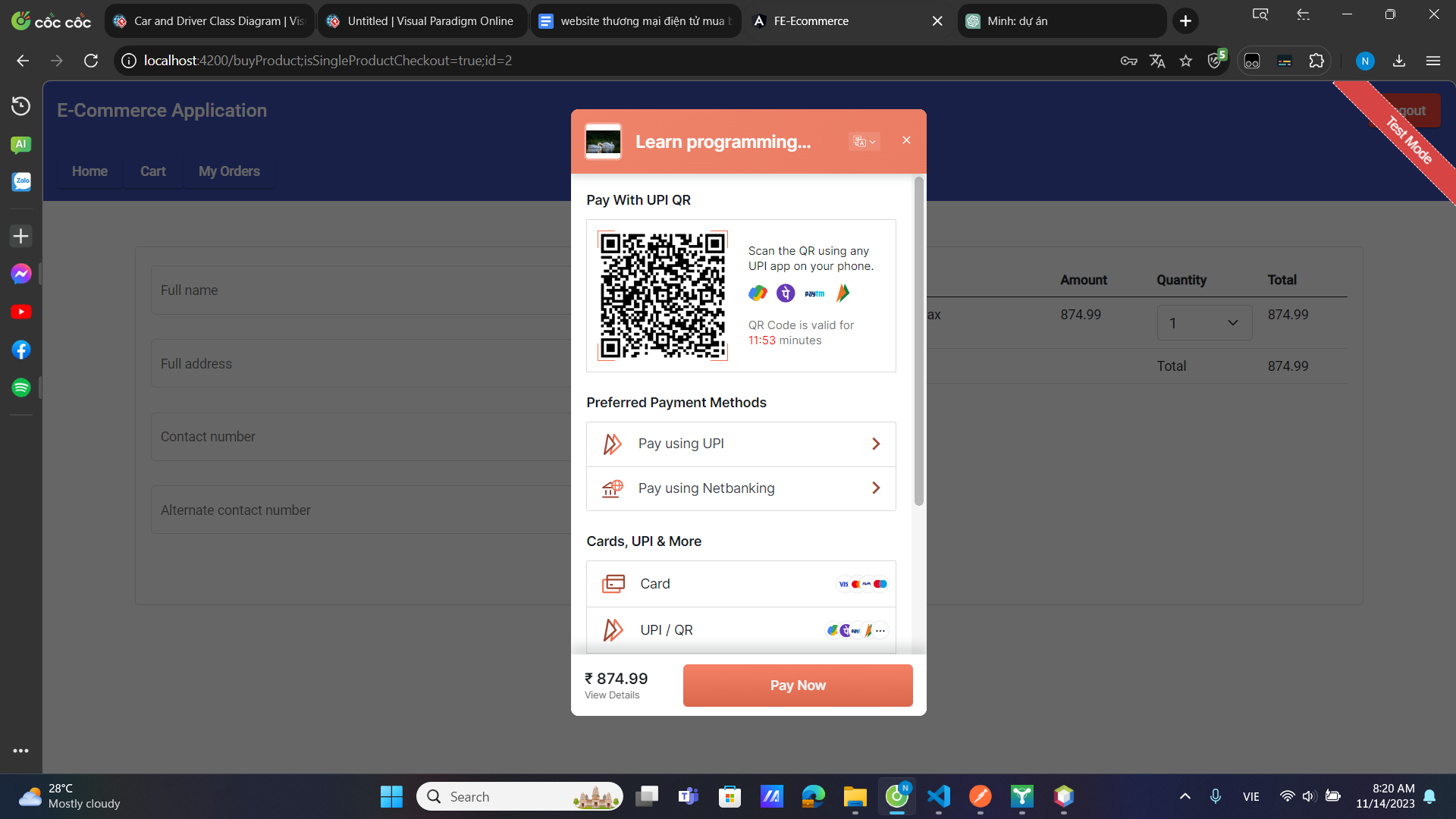
**chức năng search sản phẩm:**

****

**form mua hàng:**

****

**form thanh toán:**

****