Models/User.php:  
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

    use HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

        'city',

        'address'

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* Get the attributes that should be cast.

     \*

     \* @return array<string, string>

     \*/

    protected function casts(): array

    {

        return [

            'email\_verified\_at' => 'datetime',

            'password' => 'hashed',

        ];

    }

}

Product.php:  
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

    use HasFactory;

    protected $table = 'products';

    protected $fillable = [

        'name',

        'description',

        'price',

        'img',

        'quantity',

        'user\_id',

    ];

    public function user()

    {

        return $this->belongsTo(User::class, 'user\_id');

    }

}

Sale.php:  
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Sale extends Model

{

    use HasFactory;

    protected $table = 'sales';

    protected $fillable = ['user\_id\_buyer', 'user\_id\_seller', 'payment\_method', 'total'];

    // Relación: Un 'Sale' pertenece a un usuario comprador

    public function buyer()

    {

        return $this->belongsTo(User::class, 'user\_id\_buyer');

    }

    // Relación: Un 'Sale' pertenece a un usuario vendedor

    public function seller()

    {

        return $this->belongsTo(User::class, 'user\_id\_seller');

    }

    // Relación: Un 'Sale' tiene muchos 'DetailsSale'

    public function details()

    {

        return $this->hasMany(DetailsSale::class, 'sale\_id');

    }

}

Buy.php:  
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Buy extends Model

{

    use HasFactory;

    protected $table = 'buys';

    protected $fillable = ['user\_id\_buyer', 'user\_id\_seller','payment\_method', 'total'];

    // Relación: Un 'Buy' pertenece a un usuario comprador

    public function buyer()

    {

        return $this->belongsTo(User::class, 'user\_id\_buyer');

    }

    // Relación: Un 'Buy' pertenece a un usuario vendedor

    public function seller()

    {

        return $this->belongsTo(User::class, 'user\_id\_seller');

    }

    // Relación: Un 'Buy' tiene muchos 'DetailsBuy'

    public function details()

    {

        return $this->hasMany(DetailsBuy::class, 'buy\_id');

    }

}

DetailsBuy.php:  
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class DetailsBuy extends Model

{

    use HasFactory;

    protected $table = 'detail\_buys';

    protected $fillable = ['buy\_id', 'product\_id', 'quantity', 'total'];

    // Relación: Un 'DetailsBuy' pertenece a un 'Buy'

    public function buy()

    {

        return $this->belongsTo(Buy::class);

    }

    // Relación: Un 'DetailsBuy' pertenece a un 'Product'

    public function product()

    {

        return $this->belongsTo(Product::class, 'product\_id');

    }

}

DetailsSale.php:  
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class DetailsSale extends Model

{

    use HasFactory;

    protected $table = 'detail\_sales';

    protected $fillable = ['sale\_id', 'product\_id', 'quantity', 'total'];

    // Relación: Un 'DetailsSale' pertenece a un 'Sale'

    public function sale()

    {

        return $this->belongsTo(Sale::class);

    }

    // Relación: Un 'DetailsSale' pertenece a un 'Product'

    public function product()

    {

        return $this->belongsTo(Product::class, 'product\_id');

    }

}

CONTROLADORES:  
AuthController.php:  
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Auth;

class AuthController extends Controller

{

    public function showRegisterForm(){

        return view('register');

    }

    public function register(Request $request){

        $request->validate([

            'name' => 'required|string|max:255',

            'email' => 'required|string|email|max:255|unique:users',

            'city'=> 'required|string|max:255',

            'address'=> 'required|string|max:255',

            'password' => 'required|string|confirmed|min:8',

        ]);

        User::create([

            'name' => $request->name,

            'email' => $request->email,

            'address' => $request->address,

            'city' => $request->city,

            'password' => Hash::make($request->password),

        ]);

        if ($request->ajax()) {

            return response()->json(['success' => true]);

        }

        // return redirect()->route('login');

    }

    public function showLoginForm(){

        return view('login');

    }

    public function login(Request $request){

        $request->validate([

            'email' => 'required|string|email',

            'password' => 'required|string',

        ]);

        //Verificar que los datos esten correctos y redireccionar a home

        if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {

            // return redirect()->route('home');

            return response()->json(['success' => true]);

        }

        //De lo contrario permanecer en login y mostrar mensaje de error.

        // return redirect()->route('login')->withErrors(['email' => 'Datos incorrectos']);

        return response()->json(['errors' => ['email' => ['Las credenciales no coinciden con nuestros registros.']]], 422);

    }

    //Cerrar sesión

    public function logout()

    {

        Auth::logout();

        return redirect()->route('login');

    }

}

BuyReportController.php:  
<?php

namespace App\Http\Controllers;

use App\Models\Buy;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Services\PdfService;

class BuyReportController extends Controller

{

    protected $pdfService;

    public function \_\_construct(PdfService $pdfService)

    {

        $this->pdfService = $pdfService;

    }

    public function index()

    {

        // Obtener el ID del usuario autenticado

        $userId = Auth::id();

        // Obtener las compras realizadas por el usuario autenticado

        $buys = Buy::with(['details.product', 'buyer', 'seller']) // Usar 'details' en lugar de 'detailsBuy'

                    ->where('user\_id\_buyer', $userId) // Filtrar por el usuario autenticado (comprador)

                    ->get();

        // Pasar las compras filtradas a la vista

        return view('buy.index', compact('buys'));

    }

    public function generatePdf($buyId)

    {

        // Obtener los detalles de la compra

        $buy = Buy::with(['details.product', 'buyer', 'seller'])

            ->findOrFail($buyId);

        // Preparar datos para el PDF

        $data = [

            'buy' => $buy,

            'buyer' => $buy->buyer,

            'seller' => $buy->seller,

            'details' => $buy->details, // Usar 'details' en lugar de 'detailsBuy'

        ];

        // Generar el PDF usando el servicio

        $this->pdfService->generatePdf('buy.report', $data, "nota\_de\_compra\_{$buyId}.pdf");

    }

}

PaymentController.php:  
<?php

// PaymentController:

// Verifica nuevamente que todos los productos en el carrito sean del mismo vendedor antes de procesar el pago.

// Registra tanto la compra (Buy) como la venta (Sale) y sus detalles, actualiza el stock, y luego limpia el carrito.

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\Buy;

use App\Models\DetailsBuy;

use App\Models\Sale;

use App\Models\DetailsSale;

use App\Models\Product;

use Illuminate\Support\Facades\Auth;

class PaymentController extends Controller

{

    public function showPaymentForm()

    {

        $cart = session()->get('cart', []);

        return view('checkout.payment', compact('cart'));

    }

    public function processPayment(Request $request)

    {

        // Validar método de pago

        $request->validate([

            'payment\_method' => 'required|string|in:mercado\_pago,paypal,visa',

        ]);

        $cart = session()->get('cart', []);

        if (empty($cart)) {

            return redirect()->route('cart.index')->withErrors('Tu carrito está vacío.');

        }

        // Verificar que todos los productos en el carrito pertenecen al mismo vendedor

        $sellerId = Product::find(array\_key\_first($cart))->user\_id;

        foreach ($cart as $id => $item) {

            $product = Product::find($id);

            if ($product->user\_id !== $sellerId) {

                return redirect()->route('cart.index')->withErrors('Todos los productos en el carrito deben ser del mismo vendedor.');

            }

        }

        // Calcular el total de la compra/venta

        $total = 0;

        foreach ($cart as $item) {

            $total += $item['price'] \* $item['quantity'];

        }

        // Crear la entrada en la tabla 'buy'

        $buy = Buy::create([

            'user\_id\_seller' => $sellerId, // Se registra el vendedor

            'user\_id\_buyer' => Auth::id(),  // Se registra el comprador

            'payment\_method' => $request->payment\_method,

            'total' => $total,

        ]);

        // Crear la entrada en la tabla 'sale'

        $sale = Sale::create([

            'user\_id\_seller' => $sellerId, // Se registra el vendedor

            'user\_id\_buyer' => Auth::id(),  // Se registra el comprador

            'payment\_method' => $request->payment\_method,

            'total' => $total,

        ]);

        // Almacenar detalles de la compra y venta

        foreach ($cart as $id => $item) {

            // Insertar en 'details\_buy'

            DetailsBuy::create([

                'buy\_id' => $buy->id,

                'product\_id' => $id,

                'quantity' => $item['quantity'],

                'total' => $item['price'] \* $item['quantity'],

            ]);

            // Insertar en 'details\_sale'

            DetailsSale::create([

                'sale\_id' => $sale->id,

                'product\_id' => $id,

                'quantity' => $item['quantity'],

                'total' => $item['price'] \* $item['quantity'],

            ]);

            // Actualizar el stock del producto

            $product = Product::find($id);

            $product->quantity -= $item['quantity'];

            $product->save();

        }

        // Limpiar el carrito

        session()->forget('cart');

        return redirect()->route('home')->with('success', 'Pago realizado con éxito.');

    }

}

ProductController.php:  
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Product;

use Illuminate\Support\Facades\Auth;

class ProductController extends Controller

{

    // Mostrar todos los productos

    public function index()

    {

        $products = Product::all();

        return view('index', ['products' => $products]);

    }

    // Mostrar el formulario para crear una nueva publicación

    public function create()

    {

        return view('products.create');

    }

    // Almacenar una nueva publicación

    public function store(Request $request)

    {

        $request->validate([

            'name' => 'required|string|max:255',

            'description' => 'required|string',

            'price' => 'required|numeric',

            'quantity' => 'required|integer',

            'img' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048',

        ]);

        // Almacenar la imagen en el sistema de archivos

        $imagePath = $request->file('img')->store('products', 'public');

        Product::create([

            'name' => $request->name,

            'description' => $request->description,

            'price' => $request->price,

            'quantity' => $request->quantity,

            'img' => $imagePath,

            'user\_id' => Auth::id(), // Guardar el id del usuario publicador

        ]);

        if ($request->ajax()) {

            return response()->json(['success' => true]);

        }

        return redirect()->route('home');

    }

    // Mostrar detalles del producto

    public function show($id)

    {

        $product = Product::findOrFail($id);

        return view('products.show', compact('product'));

    }

    // Agregar producto al carrito

    public function addToCart(Request $request, $id)

{

    $product = Product::findOrFail($id);

    // Verifica si el producto tiene stock suficiente

    if ($product->quantity < $request->quantity) {

        return back()->withErrors(['message' => 'No hay suficiente stock disponible.']);

    }

    // Verifica si el usuario está intentando comprar su propio producto

    if (Auth::id() === $product->user\_id) {

        return back()->withErrors(['message' => 'No puedes comprar tu propio producto.']);

    }

    // Almacenar el artículo en el carrito en la sesión

    $cart = session()->get('cart', []);

    $cart[$id] = [

        "name" => $product->name,

        "quantity" => $request->quantity,

        "price" => $product->price,

        "img" => $product->img,

        "total" => $product->price \* $request->quantity,

    ];

    session()->put('cart', $cart);

    return redirect()->route('cart.index');

}

    // Mostrar vista del carrito de compras

    public function cart()

    {

        $cart = session()->get('cart', []);

        // Depura el carrito para asegurarte de que contiene los productos correctos

        foreach ($cart as $id => $item) {

            $product = Product::find($id);

            if ($product && $product->user\_id === Auth::id()) {

                // Eliminar productos del carrito que pertenecen al usuario

                unset($cart[$id]);

            }

        }

        session()->put('cart', $cart);

        return view('cart.index', compact('cart'));

    }

    // Eliminar artículo del carrito

    public function removeFromCart($id)

    {

        $cart = session()->get('cart');

        if (isset($cart[$id])) {

            unset($cart[$id]);

            session()->put('cart', $cart);

        }

        return redirect()->route('cart.index');

    }

    // Mostrar formulario de checkout

    public function checkout()

    {

        return view('checkout.payment');

    }

    // Administrador de productos: listar los productos del usuario autenticado// Administrador de productos: listar los productos del usuario autenticado

public function adminProduct()

{

    $products = Product::where('user\_id', Auth::id())->get();

    return view('products.adminproduct', compact('products'));

}

    // Mostrar el formulario de edición

    public function edit(Product $product)

    {

        // Verificar que el usuario tiene permiso para editar este producto

        if ($product->user\_id !== Auth::id()) {

            return redirect()->route('home')->withErrors('Unauthorized access');

        }

        return view('products.edit', compact('product'));

    }

    // Actualizar el producto

    public function update(Request $request, Product $product)

    {

        // Verificar que el usuario tiene permiso para actualizar este producto

        if ($product->user\_id !== Auth::id()) {

            return redirect()->route('home')->withErrors('Unauthorized access');

        }

        $request->validate([

            'name' => 'required|string|max:255',

            'description' => 'required|string',

            'price' => 'required|numeric',

            'quantity' => 'required|integer',

            'img' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',

        ]);

        // Obtener solo los datos necesarios del request

        $data = $request->only(['name', 'description', 'price', 'quantity']);

        // Si se proporciona una nueva imagen, guardarla en el sistema de archivos

        if ($request->hasFile('img')) {

            $data['img'] = $request->file('img')->store('products', 'public');

        }

        $product->update($data);

        if ($request->ajax()) {

            return response()->json(['success' => true]);

        }

        return redirect()->route('products.adminproduct');

    }

    // Eliminar el producto

    public function destroy(Product $product)

    {

        // Verificar que el usuario tiene permiso para eliminar este producto

        if ($product->user\_id !== Auth::id()) {

            return redirect()->route('home')->withErrors('Unauthorized access');

        }

        $product->delete();

        return redirect()->route('products.adminproduct');

    }

}

SaleReportController.php:  
<?php

namespace App\Http\Controllers;

use App\Models\Sale;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Services\PdfService;

class SaleReportController extends Controller

{

    protected $pdfService;

    public function \_\_construct(PdfService $pdfService)

    {

        $this->pdfService = $pdfService;

    }

    public function index()

    {

        // Obtener el ID del usuario autenticado

        $userId = Auth::id();

        // Obtener las ventas del usuario autenticado

        $sales = Sale::with(['details.product', 'buyer', 'seller']) // Asegúrate de que las relaciones sean correctas

                     ->where('user\_id\_seller', $userId)

                     ->get();

        // Pasar las ventas filtradas a la vista

        return view('sales.index', compact('sales'));

    }

    public function generatePdf($saleId)

    {

        // Obtener los detalles de la venta

        $sale = Sale::with(['details.product', 'buyer', 'seller'])

            ->findOrFail($saleId);

        // Preparar datos para el PDF

        $data = [

            'sale' => $sale,

            'buyer' => $sale->buyer,

            'seller' => $sale->seller,

            'details' => $sale->details,

        ];

        // Generar el PDF usando el servicio

        $this->pdfService->generatePdf('sales.report', $data, "nota\_de\_venta\_{$saleId}.pdf");

    }

}

Web.php:  
<?php

use App\Http\Controllers\BuyReportController;

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\AuthController;

use App\Http\Controllers\ProductController;

use App\Http\Controllers\PaymentController;

use App\Http\Controllers\SaleReportController;

Route::get('/', function () {

    return view('welcome');

});

//Registro e Inicio de sesión

Route::get('/register', [AuthController::class, 'showRegisterForm'])->name('register');

Route::post('/register', [AuthController::class, 'register']);

Route::get('/login', [AuthController::class, 'showLoginForm'])->name('login');

Route::post('/login', [AuthController::class, 'login']);

Route::post('/logout', [AuthController::class, 'logout'])->name('logout');

//Pagina principal

Route::get('/home', [ProductController::class, 'index'])->name('home')->middleware('auth');

//Publicar producto

Route::get('/products/create', [ProductController::class, 'create'])->name('products.create')->middleware('auth'); //Formulario de creacion

Route::post('/products', [ProductController::class, 'store'])->name('products.store')->middleware('auth'); //Procesa los datos y lo guarda en la base de datos.

//Detalle de producto

Route::get('/products/{id}', [ProductController::class, 'show'])->name('products.show');

//Ruta para agregar al carrito

Route::post('/cart/{id}', [ProductController::class, 'addToCart'])->name('cart.add');

//Mostrar carrito

Route::get('/cart', [ProductController::class, 'cart'])->name('cart.index');

//Eliminar articulo del carrito

Route::delete('/cart/{id}', [ProductController::class, 'removeFromCart'])->name('cart.remove');

//Ruta para checkout

Route::get('/checkout', [ProductController::class, 'checkout'])->name('checkout.index');

Route::get('/checkout/payment', [PaymentController::class, 'showPaymentForm'])->name('checkout.payment');

Route::post('/checkout/payment', [PaymentController::class, 'processPayment'])->name('payment.process');

//Administrador de productos

// Ruta para listar los productos del usuario

Route::get('/products', [ProductController::class, 'adminProduct'])->name('products.adminproduct')->middleware('auth');

// Rutas para editar y actualizar productos

Route::get('/products/{product}/edit', [ProductController::class, 'edit'])->name('products.edit')->middleware('auth');

Route::put('/products/{product}', [ProductController::class, 'update'])->name('products.update')->middleware('auth');

// Ruta para eliminar productos

Route::delete('/products/{product}', [ProductController::class, 'destroy'])->name('products.destroy')->middleware('auth');

Route::get('/sales', [SaleReportController::class, 'index'])->name('sales.index');

Route::get('/sales/pdf/{saleId}', [SaleReportController::class, 'generatePdf'])->name('sales.generatePdf');

Route::get('/buy', [BuyReportController::class, 'index'])->name('buy.index');

Route::get('/buy/pdf/{buyId}', [BuyReportController::class, 'generatePdf'])->name('buy.generatePdf');

Cart/index.blade.php:  
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Carrito</title>

</head>

<body>

@include('sidebar')

<h1>Carrito de Compras</h1>

@if(session('cart') && count(session('cart')) > 0)

    <table>

        <thead>

            <tr>

                <th>Imagen</th>

                <th>Nombre</th>

                <th>Cantidad</th>

                <th>Precio</th>

                <th>Total</th>

                <th>Acciones</th>

            </tr>

        </thead>

        <tbody>

        @foreach(session('cart') as $id => $item)

    @php

        $product = App\Models\Product::find($id);

    @endphp

    <tr>

        <td><img src="{{ asset('storage/' . $item['img']) }}" alt="{{ $item['name'] }}" style="width: 50px; height: auto;"></td>

        <td>{{ $item['name'] }}</td>

        <td>{{ $item['quantity'] }}</td>

        <td>${{ $item['price'] }}</td>

        <td>${{ $item['total'] }}</td>

        <td>

            <form action="{{ route('cart.remove', $id) }}" method="post">

                @csrf

                @method('DELETE')

                <button type="submit">Eliminar</button>

            </form>

        </td>

    </tr>

@endforeach

        </tbody>

    </table>

    <br>

    <a href="{{ route('checkout.payment') }}">Proceder al Pago</a>

     @else

    <p>Tu carrito está vacío.</p>

@endif

<!-- Mostrar mensajes de error -->

@if ($errors->any())

    <div>

        <ul>

            @foreach ($errors->all() as $error)

                <li>{{ $error }}</li>

            @endforeach

        </ul>

    </div>

@endif

</body>

</html>

Checkout/payment.blade.php:  
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Procesar pago</title>

</head>

<body>

@include('sidebar')

    <h1>Confirmar Pago</h1>

    <form action="{{ route('payment.process') }}" method="POST">

        @csrf

        <label for="payment\_method">Método de Pago:</label>

        <select name="payment\_method" id="payment\_method">

            <option value="mercado\_pago">Mercado Pago</option>

            <option value="paypal">PayPal</option>

            <option value="visa">Visa</option>

        </select>

        <br><br>

        <button type="submit">Confirmar Pago</button>

    </form>

    @if ($errors->any())

        <div>

            <ul>

                @foreach ($errors->all() as $error)

                    <li>{{ $error }}</li>

                @endforeach

            </ul>

        </div>

    @endif

</body>

</html>