Order system

steps

1. user add product and its quantity to cart

2.creates a payment intent for your current cart **POST /api/order/payment-intent**

3.create order user can do order الدفع عند الاستلام without online payment(stripe online payment)

4.complete payment with stripe(online payment)

Payment

⚠️ Important: The publishable key (pk_test_...)

is used on the frontend

publishable key = pk_test_51ST4ywRxntGfyOLIilQiBo8gBN5n0DDXcchPZ3LouSTHw0xvp6QD4NeEJXW3KwM6uM63TVoc6jmUOGXAw6tgedHT00cVhREwD5

you can also make order without payment on strip by make user pay cash on delivery like :

POST /api/order

```
{
"deliveryMethodId": 1,
"paymentMethod": 2,   //this cash on delivery method doesn't need online payment
"shippingAddress": { "fName": "John", "lName": "Doe", "city": "Cairo", "street": "123 Main St", "country": "Egypt" }

}
```

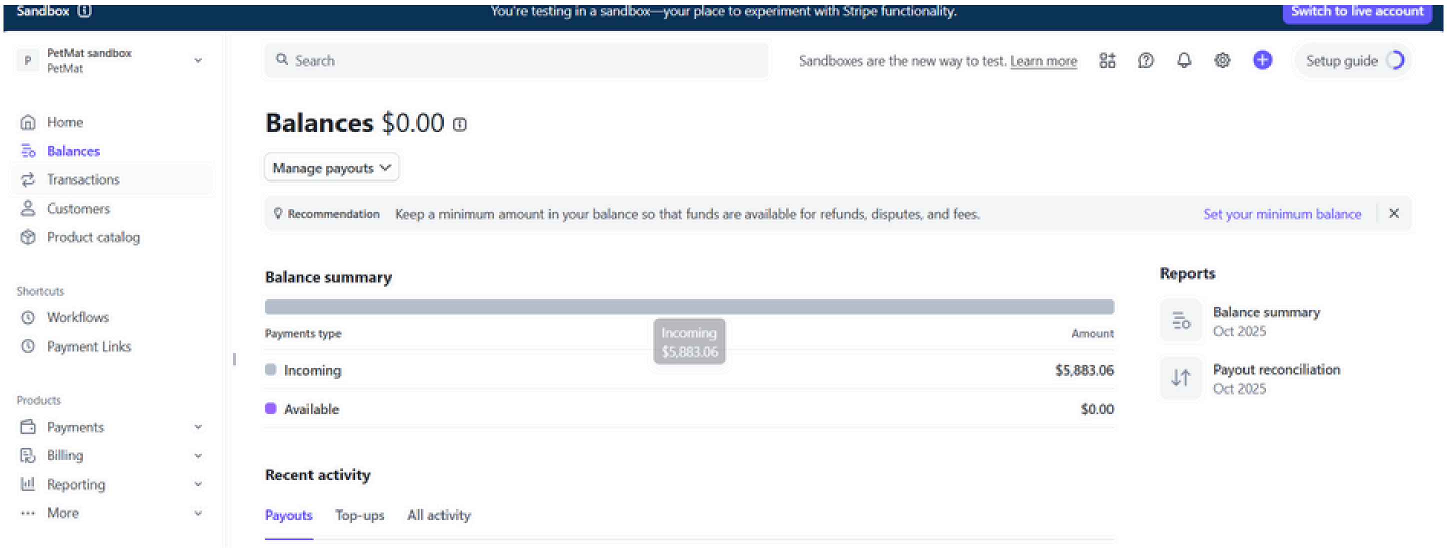
----

paymentMethod: 2 this cash

paymentMethod : 1 this visa like strip getway

----

after payment on strip you can check on our strip account of pet mat



You're testing in a sandbox—your place to experiment with Stripe functionality.

Sandboxes are the new way to test. Learn more

Balances $0.00

Recommendation Keep a minimum amount in your balance so that funds are available for refunds, disputes, and fees.

Balance summary

Incoming $5,883.06

Payments type	Amount
Incoming	$5,883.06
Available	$0.00

incoming value is the payment happened in our strip account

Admin

Admin Coupon /api/admincoupon — Auth: Admin

Get all coupons — GET /api/admincoupon

Auth: Required (Admin)

Responses: • 200 -> CouponListDto

- Get coupon by id — GET /api/admincoupon/{id}

Auth: Required (Admin)

Path: • id (int)

Responses: • 200 -> CouponDto

• 404 -> not found

Get coupon by code — GET /api/admincoupon/code/{code}

Auth: Required (Admin)

Path: • code (string)

Responses: • 200 -> CouponDto

• 404 -> not found

Create coupon — POST /api/admincoupon

Auth: Required (Admin) Body: application/json -> AddCouponDto

• Name (string, max 200) — required

• Code (string, max 50) — required

• Rate (decimal, >=0) — required

• IsPercentage (bool) — required

• ExpiresAt (datetime?) — optional

• MinOrderAmount (decimal) — optional

Responses: • 200 -> CouponOperationResponseDto { Success, Message, CouponId }

• 400 -> validation / invalid operation

Update coupon — PUT /api/admincoupon/{id}

Auth: Required (Admin) Body: application/json -> UpdateCouponDto (fields optional)

Responses: • 200 -> CouponOperationResponseDto

• 404 -> not found

Delete coupon — DELETE /api/admincoupon/{id}

Auth: Required (Admin)

Path: • id (int)

Responses: • 200 -> CouponOperationResponseDto

• 404 -> not found

Admin Delivery Method /api/admindeliverymethod — Auth: Admin (GETs AllowAnonymous)

Get all delivery methods — GET /api/admindeliverymethod

Auth: Optional (AllowAnonymous)

Responses: • 200 -> DeliveryMethodListDto

Get delivery method by id — GET /api/admindeliverymethod/{id}

Auth: Optional (AllowAnonymous)

Path: • id (int)

Responses: • 200 -> DeliveryMethodDto

• 404 -> not found

Create delivery method — POST /api/admindeliverymethod

Auth: Required (Admin)

Body: application/json -> AddDeliveryMethodDto

• ShortName (string, max 100) — required

• Description (string) — optional

• DeliveryTime (string, max 100) — required

• Cost (decimal, >=0) — required

Responses: • 200 -> DeliveryMethodOperationResponseDto

• 400 -> validation

Update delivery method — PUT /api/admindeliverymethod/{id}

Auth: Required (Admin)

Body: application/json -> UpdateDeliveryMethodDto

Responses: • 200 -> DeliveryMethodOperationResponseDto

• 404 -> not found

Delete delivery method — DELETE /api/admindeliverymethod/{id}

Auth: Required (Admin)

Path: • id (int)

Responses: • 200 -> DeliveryMethodOperationResponseDto

• 400 -> invalid operation (used in orders)

• 404 -> not found

Admin Order /api/adminorder — Auth: Admin

Get all orders — GET /api/adminorder?pageIndex=1&pageSize=10&status={status}

Auth: Required (Admin)

Query: • pageIndex (int, default 1)

• pageSize (int, default 10)

• status (string, optional)

Responses: • 200 -> PaginationResponse<OrderDto>

• 400 -> invalid paging/filter values

Get order by id — GET /api/adminorder/{orderId}

Auth: Required (Admin)

Path: • orderId (int)

Responses: • 200 -> OrderDto

• 404 -> not found

Update order status — PUT /api/adminorder/{orderId}/status

Auth: Required (Admin)

Body: application/json -> UpdateOrderStatusDto

• Status (enum: OrderStatus) — required

Responses: • 200 -> OrderOperationResponseDto

• 400 -> validation

• 404 -> not found

Delete order (restore stock) — DELETE /api/adminorder/{orderId}

Auth: Required (Admin)

Path: • orderId (int)

Responses: • 200 -> OrderOperationResponseDto

• 404 -> not found

Admin Product /api/adminproduct — Auth: Admin Product management:

Add product — POST /api/adminproduct/product

Auth: Required (Admin) Body: multipart/form-data -> AddProductDto

• Name (string, max 200) — required

• Description (string) — optional

• PictureFile (file) — required

• Price (decimal, >=0) — required

• Stock (int, >=0) — required

• BrandId (int) — required

• TypeId (int) — required

• SpeciesId (int?, optional)

• NutritionalInfo (string) — optional

• Ingredients (string) — optional

• ExpiryDate (datetime?) — optional

Responses: • 200 -> ProductOperationResponseDto { Success, Message, ProductId }

• 400 -> validation

• 404 -> brand/type/species not found

Update product — PUT /api/adminproduct/product/{id}

Auth: Required (Admin) Body: multipart/form-data -> UpdateProductDto (all fields optional)

Responses: • 200 -> ProductOperationResponseDto

• 400 -> validation

• 404 -> not found

Delete product — DELETE /api/adminproduct/product/{id}

Auth: Required (Admin)

Path: • id (int)

Responses: • 200 -> ProductOperationResponseDto

• 400 -> invalid operation (in cart/order)

• 404 -> not found

Activate product — PUT /api/adminproduct/product/{id}/activate

Auth: Required (Admin)

Responses: • 200 -> ProductOperationResponseDto

• 404 -> not found

Deactivate product — PUT /api/adminproduct/product/{id}/deactivate

Auth: Required (Admin)

Responses: • 200 -> ProductOperationResponseDto

• 404 -> not found

Brand management:

Add brand — POST /api/adminproduct/brand

Auth: Required (Admin) Body: multipart/form-data -> AddProductBrandDto

• Name (string, max 200) — required

• Description (string) — optional

• LogoFile (file) — optional

Responses: • 200 -> BrandOperationResponseDto

• 400 -> validation

Update brand — PUT /api/adminproduct/brand/{id}

Auth: Required (Admin) Body: multipart/form-data -> UpdateProductBrandDto

Responses: • 200 -> BrandOperationResponseDto

• 400/404 -> error

Delete brand — DELETE /api/adminproduct/brand/{id}

Auth: Required (Admin)

Responses: • 200 -> BrandOperationResponseDto

• 400 -> invalid operation (brand has products)

• 404 -> not found

Type management:

Add type — POST /api/adminproduct/type

Auth: Required (Admin) Body: application/json -> AddProductTypeDto

• Name (string, max 100) — required

• Description (string) — optional

Responses: • 200 -> TypeOperationResponseDto

Update type — PUT /api/adminproduct/type/{id}

Auth: Required (Admin) Body: application/json -> UpdateProductTypeDto

Responses: • 200 -> TypeOperationResponseDto

Delete type — DELETE /api/adminproduct/type/{id}

Auth: Required (Admin)

Responses: • 200 -> TypeOperationResponseDto

• 400 -> invalid operation (type has products)

• 404 -> not found

User

Cart /api/cart — Auth: Required

Get current user's cart — GET /api/cart

Auth: Required

Responses: • 200 -> CartDto { Id, UserId, CouponCode, DiscountAmount, SubTotal, Total, Items[], LastUpdated }

Add product to cart — POST /api/cart/add

Auth: Required Body: application/json -> AddToCartDto

• ProductId (int) — required

• Quantity (int >=1) — required

Responses: • 200 -> CartOperationResponseDto { Success, Message, Cart }

• 400 -> invalid operation (insufficient stock)

• 404 -> product not found

Update cart item quantity — PUT /api/cart/item/{cartItemId}

Auth: Required Body: application/json -> UpdateCartItemDto

• Quantity (int >=1) — required

Responses: • 200 -> CartOperationResponseDto

• 400 -> invalid operation (insufficient stock)

• 404 -> cart/cart item not found

Remove item from cart — DELETE /api/cart/item/{cartItemId}

Auth: Required Path: • cartItemId (int)

Responses: • 200 -> CartOperationResponseDto

• 404 -> cart/cart item not found

Clear cart — DELETE /api/cart/clear

Auth: Required

Responses: • 200 -> CartOperationResponseDto

Apply coupon to cart — POST /api/cart/coupon

Auth: Required Body: application/json -> ApplyCouponDto

• CouponCode (string, max 50) — required

Responses: • 200 -> CartOperationResponseDto (Cart updated)

• 400 -> invalid coupon conditions / cart empty

• 404 -> coupon not found

Remove coupon from cart — DELETE /api/cart/coupon

Auth: Required

Responses: • 200 -> CartOperationResponseDto

• 404 -> cart not found

Order /api/order — Auth: Required

Create order (online or COD) — POST /api/order

Auth: Required Body: application/json -> CreateOrderDto

• DeliveryMethodId (int) — required

• ShippingAddress (OrderAddressDto) — required (FName, LName, City, Street, Country)

• PaymentMethod (enum: Online | CashOnDelivery) — required

Responses: • 200 -> OrderDto (created order)

• 400 -> invalid operation (cart empty, insufficient stock, missing payment intent)

• 404 -> delivery method not found

Get order by id (user) — GET /api/order/{orderId}

Auth: Required Path: • orderId (int)

Responses: • 200 -> OrderDto

• 404 -> not found

Get my orders — GET /api/order/my-orders

Auth: Required

Responses: • 200 -> OrderListDto

• Validate order exists for payment — GET /api/order/validate-payment/{paymentIntentId}

Auth: Required Path: • paymentIntentId (string)

Responses: • 200 -> { exists: true, message } or 404 -> not found / invalid

Create / update payment intent (client secret) — POST /api/order/payment-intent

Auth: Required

Responses: • 200 -> PaymentIntentResponseDto { PaymentIntentId, ClientSecret, Amount, Currency }

• 400 -> invalid operation

Product /api/product — Auth: Optional (public)

Get all products — GET /api/product?PageIndex=1&pageSize=10&BrandId=&TypeId=&SpeciesId=&MinPrice=&MaxPrice=&Search=&InStock=&SortBy=

Auth: Optional (AllowAnonymous) Query: ProductFilterParams

• PageIndex (int, default 1)

• PageSize (int, default 10)

• BrandId (int?)

• TypeId (int?)

• SpeciesId (int?)

• MinPrice / MaxPrice (decimal?)

• Search (string)

• InStock (bool?)

• SortBy (string: price_asc|price_desc|name|newest)

Responses: • 200 -> PaginationResponse<ProductDto>

• 400 -> invalid paging/filter values

Get product by id — GET /api/product/{id}

Auth: Optional (AllowAnonymous) Path: • id (int)

Responses: • 200 -> ProductDto

• 404 -> not found

Get all brands — GET /api/product/brands

Auth: Optional (AllowAnonymous)

Responses: • 200 -> ProductBrandListDto

Get all types — GET /api/product/types

Auth: Optional (AllowAnonymous)

Responses: • 200 -> ProductTypeListDto

Get delivery methods (public) — GET /api/order/delivery-methods

Auth: Optional (AllowAnonymous)

Responses: • 200 -> DeliveryMethodListDto

Quick DTO / enum references (frontend)

- Payment: PaymentIntentResponseDto { PaymentIntentId, ClientSecret, Amount, Currency }
- Enums: PaymentMethod (Online | CashOnDelivery)