Group 4 In Database System Lecture

RT-MART E-COMMERCE PLATFORM

https://aps.ntut.edu.tw/course/tw/Curr.jsp?format=-2&code=5902206

hyujun940529@gmail.com

106344 No. 1, Sec. 3,
Zhongxiao E. Rd., Da'an Dist., Taipei City

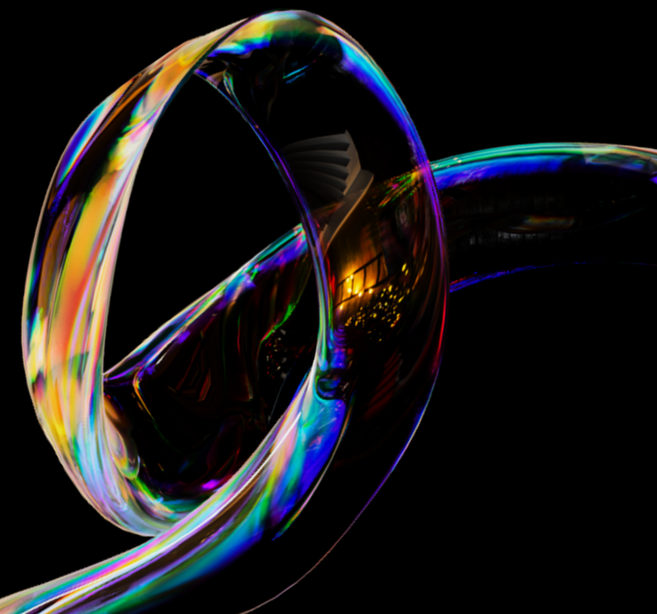
TABLE OF CONTENTS

INTRODUCTION

線上電商系統「大潤發電商平臺」

支援買家、賣家與管理員三方角色的操作需求。

買家 能夠進行商品的搜尋、瀏覽與購買；
賣家 能夠維護商店資訊並管理商品；
管理員 能夠掌控與維護整體系統的正常運作。



TECHNOLOGY STACK

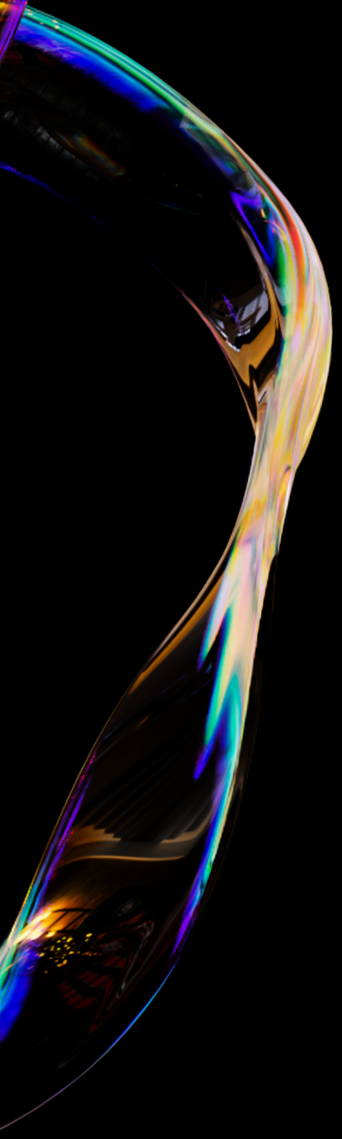
FRAMEWORK LAYER

React

nest

RUNTIME ENVIRONMENT

node JS



TECHNOLOGY STACK

DBMS

MariaDB

TYPEORM

TECHNOLOGY STACK

DEVELOPMENT TOOLS

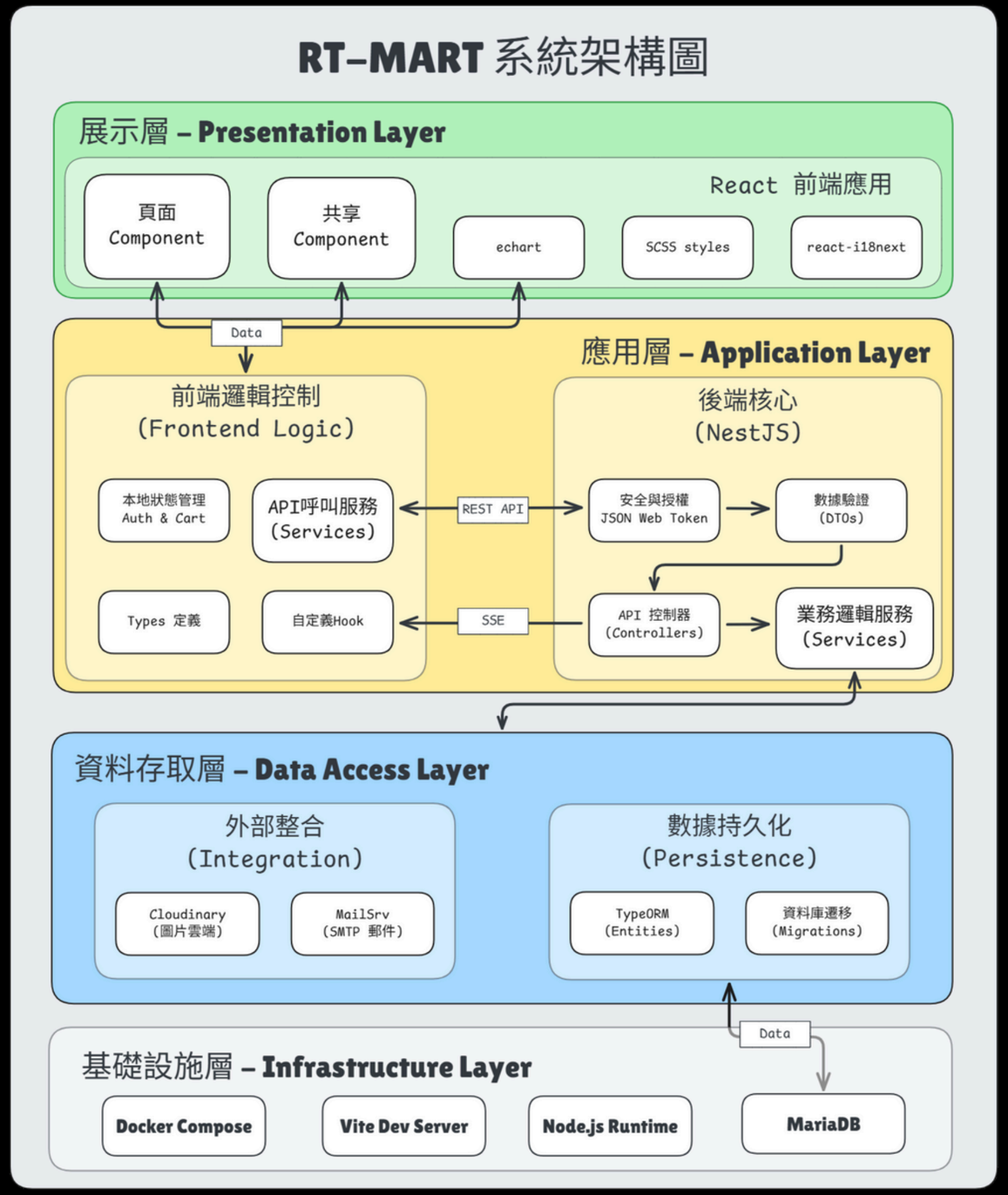
visual studio code

docker

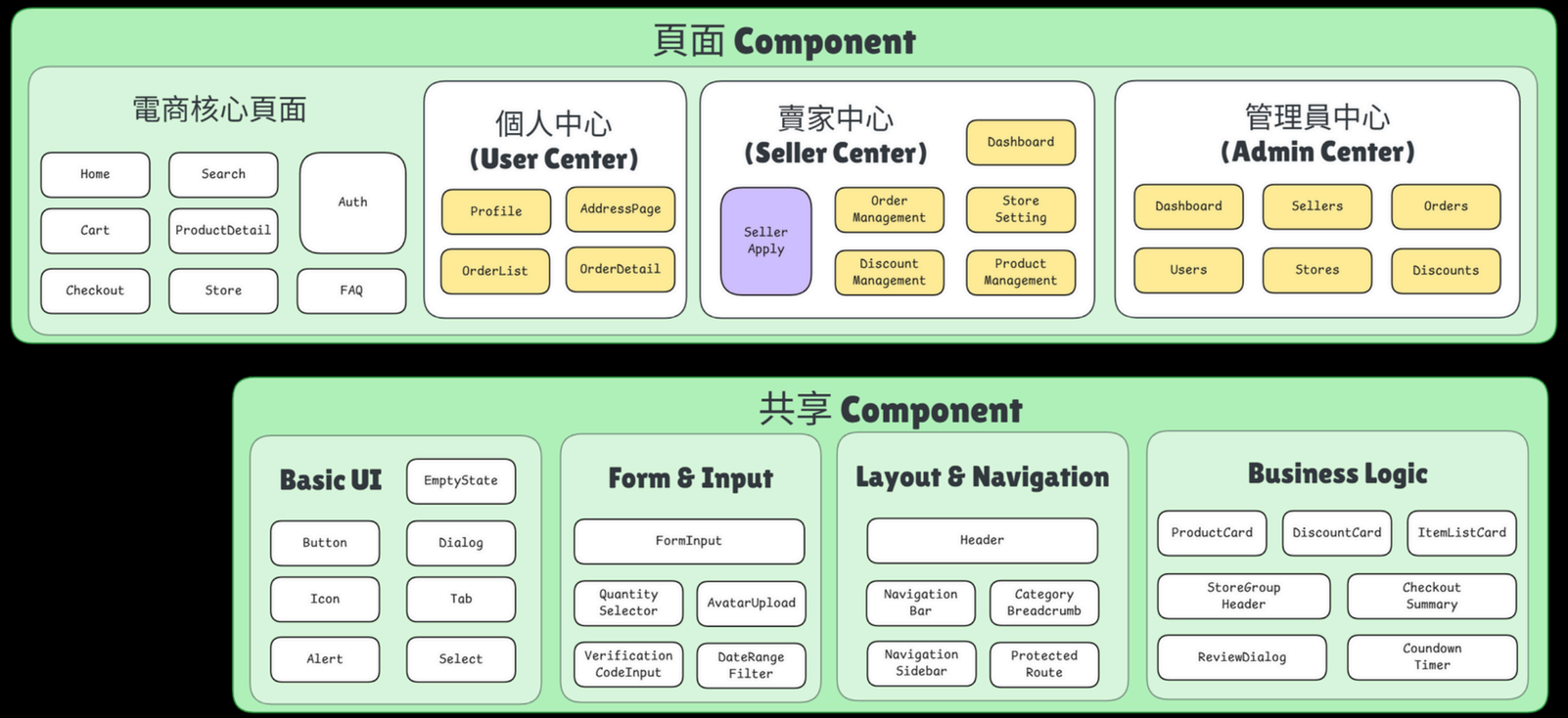
ChatGPT Gemini Claude

Github

SYSTEM ARCHITECTURE

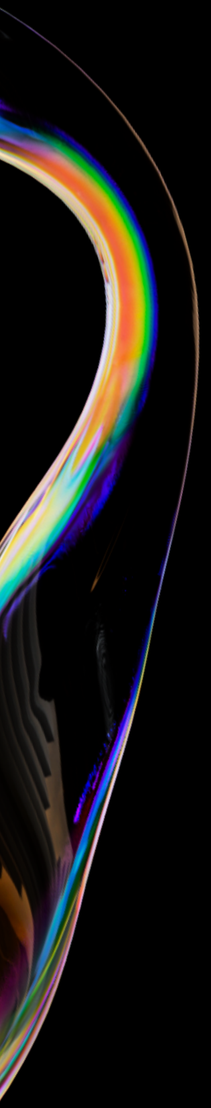


SYSTEM ARCHITECTURE



SYSTEM ARCHITECTURE

API LIST



API 總數: 146個
統計時間: 2026年1月7日

Module	Method	Path	Description
Users	POST	/users	創建用戶
Users	PATCH	/users/me	用戶更新個人資料
Users	DELETE	/users/me	用戶註銷個人帳號

Users	GET	/users	管理員分頁查詢所有用戶
Users	GET	/users/deleted	管理員查詢已刪除用戶
Users	GET	/users/:id	查詢單一用戶資料
Users	PATCH	/users/:id	管理員更新用戶資料
Users	DELETE	/users/:id	管理員軟刪除用戶
Users	POST	/users/:id/restore	管理員還原已刪除用戶
Users	POST	/users/:id/suspend	管理員停權用戶
Users	POST	/users/:id/restore-suspended	管理員取消停權
Users	DELETE	/users/:id/permanent	管理員永久刪除用戶
Users	GET	/users/test/health	用戶模組健康檢查

Sellers	POST	/sellers	買家申請成為賣家
Sellers	GET	/sellers	管理員查詢所有賣家申請
Sellers	GET	/sellers/dashboard	賣家儀表板數據
Sellers	GET	/sellers/sales-report	賣家下載銷售報表 (CSV)
Sellers	GET	/sellers/:sellerId	管理員查詢特定賣家資料
Sellers	PATCH	/sellers	賣家更新個人賣家資訊
Sellers	POST	/sellers/:sellerId/verify	管理員審核通過賣家
Sellers	POST	/sellers/:sellerId/reject	管理員拒絕賣家申請
Sellers	DELETE	/sellers/:sellerId	管理員刪除賣家身份
Sellers	GET	/sellers/test/health	賣家模組健康檢查

Admin	GET	/admin/dashboard/stats	管理員儀表板統計

Auth	POST	/auth/register	用戶註冊
Auth	POST	/auth/register/send-code	發送郵件驗證碼
Auth	POST	/auth/register/verify	驗證代碼並完成註冊
Auth	POST	/auth/login	用戶登入
Auth	POST	/auth/refresh	刷新 Access Token
Auth	GET	/auth/profile	取得當前用戶資訊
Auth	POST	/auth/logout	用戶登出
Auth	GET	/auth/test/health	認證模組健康檢查

AuditLogs	POST	/audit-logs	內部記錄審計日誌
AuditLogs	GET	/audit-logs	管理員查詢審計日誌
AuditLogs	GET	/audit-logs/statistics	取得系統操作統計
AuditLogs	GET	/audit-logs/event/:eventId	根據事件 ID 查詢日誌
AuditLogs	GET	/audit-logs/request/:requestId	根據請求 ID 追蹤操作
AuditLogs	GET	/audit-logs/verify/:id	驗證日誌數據完整性
AuditLogs	GET	/audit-logs/user/:userId	查詢特定用戶操作日誌
AuditLogs	GET	/audit-logs/entity/:tableName/:recordId	查詢實體變更歷史
AuditLogs	GET	/audit-logs/test/health	稽核模組健康檢查
AuditLogs	GET	/audit-logs/:id	查詢單一日誌詳情

Products	GET	/products/storefront	前台產品列表查詢
Products	GET	/products/storefront/:id	前台產品詳情查詢
Products	GET	/products	查詢所有產品
Products	GET	/products/:id	查詢單一產品詳情
Products	GET	/products/test/health	產品模組健康檢查

Products	GET	/products/admin	管理員查詢產品列表
Products	GET	/products/admin/:id	管理員查詢特定產品詳情
Products	DELETE	/products/:id (Admin)	管理員強制刪除產品

Products	POST	/products	賣家上架產品
Products	GET	/products/seller	賣家查詢自家產品列表
Products	GET	/products/seller/:id	賣家查詢特定產品詳情
Products	PATCH	/products/:id	賣家更新產品資料
Products	POST	/products/:id/images	賣家為產品新增圖片
Products	PATCH	/products/:id/images/sort	賣家調整產品圖片排序
Products	DELETE	/products/:id/images/:imageId	賣家刪除產品圖片
Products	DELETE	/products/:id (Seller)	賣家下架/刪除產品

ProductTypes	POST	/product-types	管理員創建產品分類
ProductTypes	GET	/product-types	查詢所有分類列表
ProductTypes	GET	/product-types/admin	管理員查詢全部分類
ProductTypes	GET	/product-types/:id/descendant-ids	查詢分類及其子分類 ID
ProductTypes	GET	/product-types/:id	查詢特定分類詳情
ProductTypes	GET	/product-types/admin/:id	管理員查詢特定分類詳情
ProductTypes	PATCH	/product-types/:id	管理員修改分類
ProductTypes	DELETE	/product-types/:id	管理員刪除分類
ProductTypes	GET	/product-types/test/health	分類模組健康檢查

Inventory	GET	/inventory/product/:productId	查詢產品庫存資訊
Inventory	GET	/inventory/product/:productId/quantity	查詢產品可用庫存
Inventory	PATCH	/inventory/product/:productId	賣家更新庫存數量
Inventory	GET	/inventory/test/health	庫存模組健康檢查

Orders	POST	/orders	創建新訂單
Orders	POST	/orders/from-snapshot	從購物車快照創建訂單
Orders	GET	/orders	買家查詢個人訂單列表
Orders	GET	/orders/:id	買家查詢訂單詳情
Orders	PATCH	/orders/:id/status	買家更新訂單狀態
Orders	POST	/orders/:id/cancel	買家取消訂單

Orders	GET	/orders/seller/orders	賣家查詢店鋪訂單
Orders	GET	/orders/seller/orders/:id	賣家查詢店鋪特定訂單
Orders	PATCH	/orders/seller/orders/:id/status	賣家更新訂單出貨狀態

Orders	GET	/orders/admin/all	管理員查詢全平台訂單
Orders	GET	/orders/admin/:id	管理員查詢特定訂單
Orders	POST	/orders/admin/:id/cancel	管理員強制取消訂單
Orders	PATCH	/orders/admin/:id/status	管理員更新訂單狀態
Orders	GET	/orders/admin/anomalies	管理員查詢異常訂單
Orders	GET	/orders/test/health	訂單模組健康檢查

Discounts	GET	/discounts	查詢優惠券列表
Discounts	GET	/discounts/available	根據購物車查詢可用優惠券
Discounts	GET	/discounts/recommended	取得推薦優惠券
Discounts	GET	/discounts/:id	查詢特定優惠券詳情
Discounts	GET	/discounts/code/:code	根據代碼查詢優惠券
Discounts	POST	/discounts/validate/:code	驗證優惠券有效性
Discounts	GET	/discounts/test/health	優惠券模組健康檢查

Discounts	POST	/discounts	賣家創建店鋪優惠券
Discounts	PATCH	/discounts/:id	賣家更新店鋪優惠券
Discounts	DELETE	/discounts/:id (Seller)	賣家刪除優惠券

Discounts	POST	/discounts/admin	管理員創建平台優惠券
Discounts	PATCH	/discounts/admin/:id	管理員更新平台優惠券
Discounts	DELETE	/discounts/:id (Admin)	管理員刪除平台優惠券

Stores	GET	/stores	查詢所有公開店鋪
Stores	GET	/stores/me	賣家查詢個人店鋪資料
Stores	GET	/stores/:storeId	查詢特定店鋪詳情
Stores	PATCH	/stores	賣家更新店鋪資訊
Stores	POST	/stores/:storeId/restore	管理員還原已關閉店鋪
Stores	PATCH	/stores/:storeId	管理員更新店鋪資訊
Stores	DELETE	/stores	賣家自行關閉店鋪
Stores	DELETE	/stores/:storeId	管理員軟刪除店鋪
Stores	DELETE	/stores/:storeId/permanent	管理員永久刪除店鋪
Stores	GET	/stores/test/health	店鋪模組健康檢查

ShippingAddresses	POST	/shipping-addresses	用戶新增收貨地址
ShippingAddresses	GET	/shipping-addresses	用戶查詢所有收貨地址
ShippingAddresses	GET	/shipping-addresses/default	用戶查詢預設收貨地址
ShippingAddresses	GET	/shipping-addresses/:id	查詢特定地址詳情
ShippingAddresses	PATCH	/shipping-addresses/:id	更新收貨地址
ShippingAddresses	POST	/shipping-addresses/:id/set-default	設為預設地址
ShippingAddresses	DELETE	/shipping-addresses/:id	刪除收貨地址
ShippingAddresses	GET	/shipping-addresses/test/health	地址模組健康檢查

Carts	GET	/carts	查詢個人購物車內容
Carts	GET	/carts/summary	取得購物車總結數據
Carts	POST	/carts/items	將產品加入購物車
Carts	POST	/carts/items/to-history	將選中項目移至歷史
Carts	PATCH	/carts/items/batch	批量更新購物車項目
Carts	PATCH	/carts/items/:cartItemId	更新單一購物車項目
Carts	DELETE	/carts/items/:cartItemId	從購物車移除項目
Carts	DELETE	/carts/selected	移除購物車中選中的項目
Carts	DELETE	/carts	清空購物車
Carts	GET	/carts/test/health	購物車模組健康檢查

CartHistory	GET	/cart-history	用戶查詢購物車快照歷史
CartHistory	GET	/cart-history/:id	查詢特定購物歷史詳情
CartHistory	POST	/cart-history/:id/restore	將歷史快照內容還原至購物車
CartHistory	POST	/cart-history/:id/to-order	從歷史快照直接建立訂單
CartHistory	GET	/cart-history/test/health	歷史紀錄模組健康檢查

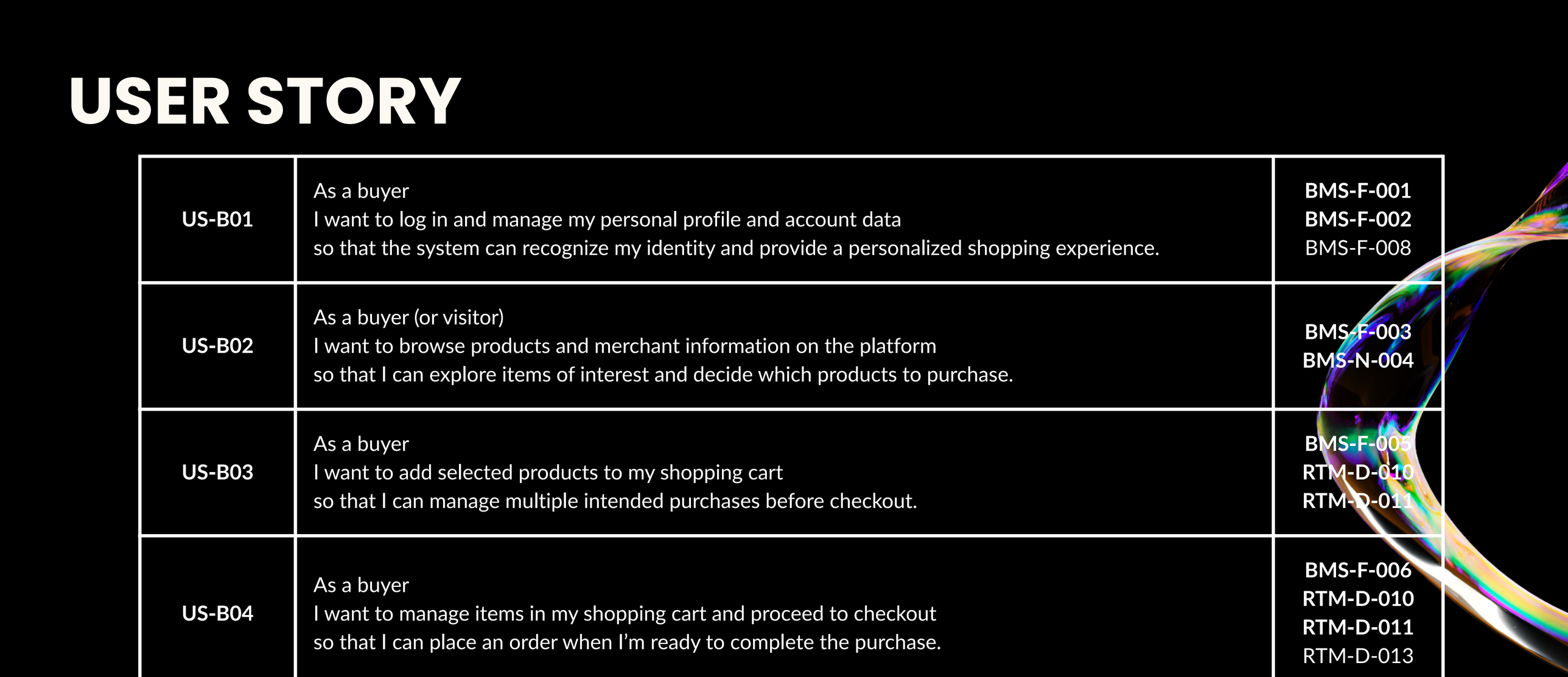
SSE	GET	/sse/events	建立 SSE 事件流連線
SSE	GET	/sse/health	SSE 服務健康監控連線

EmailVerification	POST	/email-verification/send	請求發送驗證碼
EmailVerification	POST	/email-verification/verify	提交驗證碼進行驗證
EmailVerification	POST	/email-verification/resend	重新發送驗證碼

Review	POST	/review	用戶發表評價
Review	GET	/review	查詢所有評價列表
Review	GET	/review/statistics/:productId	取得產品評價統計
Review	GET	/review/:id	查詢單一評價內容
Review	PATCH	/review/:id	用戶修改評價
Review	DELETE	/review/:id	刪除評價

USER STORY

US-B01	As a buyer I want to log in and manage my personal profile and account data so that the system can recognize my identity and provide a personalized shopping experience.	**BMS-F-001** **BMS-F-002** BMS-F-008
US-B02	As a buyer (or visitor) I want to browse products and merchant information on the platform so that I can explore items of interest and decide which products to purchase.	**BMS-F-003** **BMS-N-004**
US-B03	As a buyer I want to add selected products to my shopping cart so that I can manage multiple intended purchases before checkout.	**BMS-F-005** **RTM-D-010** **RTM-D-011**
US-B04	As a buyer I want to manage items in my shopping cart and proceed to checkout so that I can place an order when I'm ready to complete the purchase.	**BMS-F-006** **RTM-D-010** **RTM-D-011** RTM-D-013
US-B05	As a buyer I want to track and view my current and past orders so that I can know the delivery status and keep a record of my purchase history.	**BMS-F-007** **RTM-D-013** **RTM-D-014**

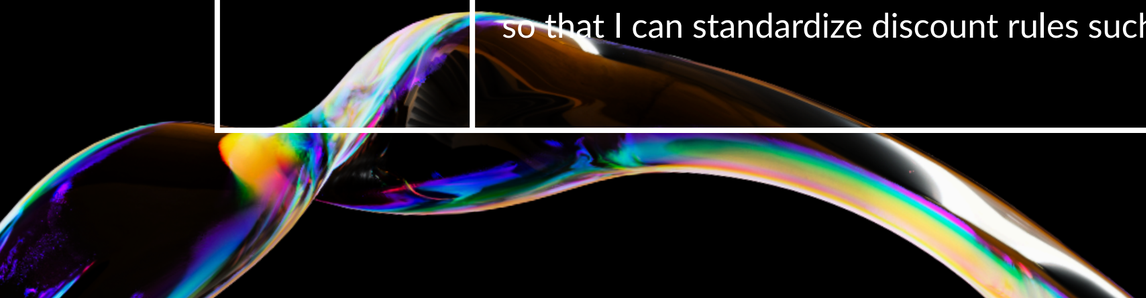


USER STORY

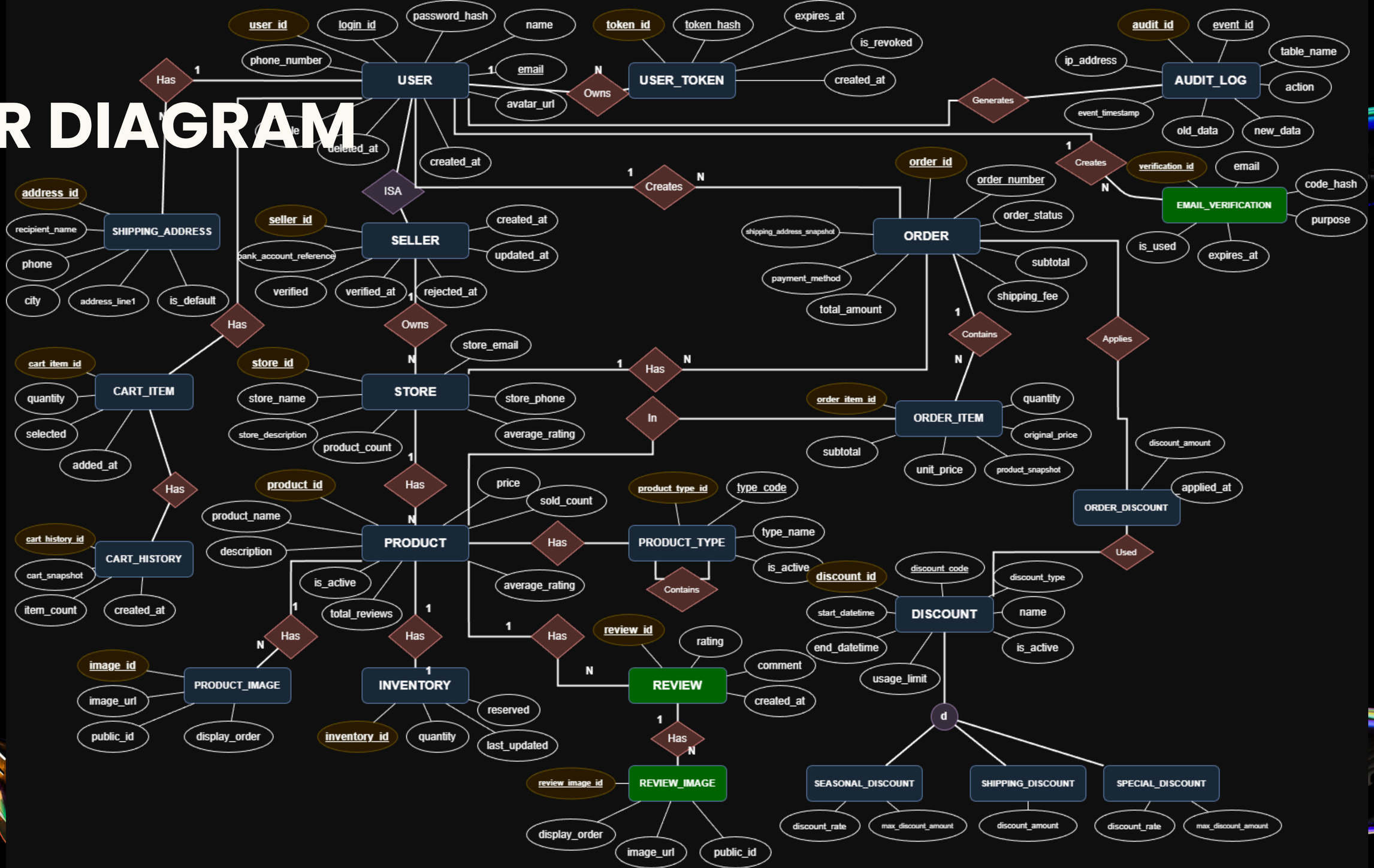
US-S01	As a seller I want to register for a seller account through the seller profile subsystem and wait for admin verification so that I can gain authorization to operate my store on the platform.	**BMS-F-004** **RTM-D-002** **SMS-N-002**
US-S02	As a seller I want to manage all products and product categories within my store so that I can easily maintain product listings, update information, and organize my inventory effectively.	**SMS-F-001** **RTM-D-006** **RTM-D-007** **RTM-D-009**
US-S03	As a seller I want to review incoming orders, accept or reject them so that I can review and manage customer orders efficiently.	**SMS-F-002** **RTM-D-013**
US-S04	As a seller I want to access and review sales reports generated from my store’s transactions so that I can analyze business performance and make informed operational decisions.	**SMS-F-004** **RTM-D-013** **RTM-D-014**
US-S05	As a seller I want to create and manage special discounts for promotional events so that I can attract more customers and increase sales through marketing activities.	**SMS-F-003** **RTM-D-015** **RTM-D-018**

USER STORY

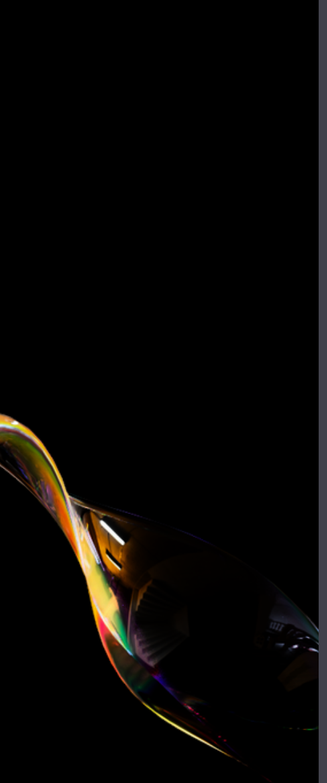
US-A01	As an admin I want to review buyers' personal information so that I can ensure the authenticity of all registered users on the platform and suspend accounts if violations occur.	**AMS-F-001** **RTM-D-001**
US-A02	As an admin I want to view and verify seller information and related business data so that I can manage seller permissions and suspend accounts if violations occur.	**AMS-F-002** **RTM-D-002**
US-A03	As an admin I want to assist sellers with transaction supervision and maintain overall order reports so that I can monitor the system's trading health and resolve disputes effectively.	**AMS-F-003** **RTM-D-013**
US-A04	As an admin I want to define and manage platform-wide promotional and discount policies so that I can standardize discount rules such as shipping discounts or seasonal sales across all stores.	**AMS-F-004** **AMS-F-005** **RTM-D-015** **RTM-D-016** **RTM-D-017**

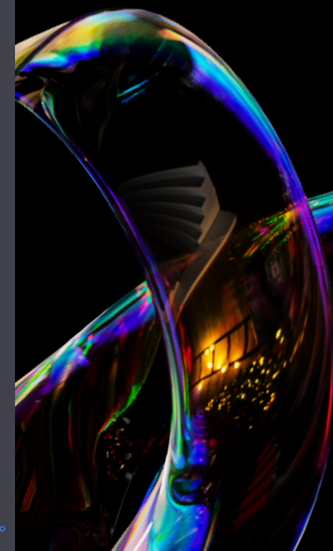


ER DIAGRAM


USER
user_id
login_id
password_hash
name
phone_number
email
avatar_url
deleted_at
created_at
Has
ISA
Owns
USER_TOKEN
token_id
token_hash
expires_at
is_revoked
created_at
AUDIT_LOG
audit_id
event_id
table_name
ip_address
action
event_timestamp
old_data
new_data
Generates
Creates
EMAIL_VERIFICATION
verification_id
email
code_hash
purpose
is_used
expires_at
SHIPPING_ADDRESS
address_id
recipient_name
phone
city
address_line1
is_default
SELLER
seller_id
bank_account_reference
verified
verified_at
rejected_at
created_at
updated_at
ORDER
order_id
order_number
order_status
shipping_address_snapshot
payment_method
total_amount
subtotal
shipping_fee
Contains
Applies
CART_ITEM
cart_item_id
quantity
selected
added_at
STORE
store_id
store_name
store_description
product_count
store_email
store_phone
average_rating
ORDER_ITEM
order_item_id
quantity
original_price
subtotal
unit_price
product_snapshot
ORDER_DISCOUNT
discount_amount
applied_at
CART_HISTORY
cart_history_id
cart_snapshot
item_count
created_at
PRODUCT
product_id
product_name
description
price
sold_count
is_active
total_reviews
average_rating
PRODUCT_TYPE
product_type_id
type_code
type_name
is_active
Contains
DISCOUNT
discount_id
discount_code
discount_type
name
is_active
start_datetime
end_datetime
usage_limit
Used
PRODUCT_IMAGE
image_id
image_url
public_id
display_order
INVENTORY
inventory_id
quantity
reserved
last_updated
REVIEW
review_id
rating
comment
created_at
REVIEW_IMAGE
review_image_id
display_order
image_url
public_id
d
SEASONAL_DISCOUNT
discount_rate
max_discount_amount
SHIPPING_DISCOUNT
discount_amount
SPECIAL_DISCOUNT
discount_rate
max_discount_amount

SCHEMA





EmailVerification

verification_id	bigint
email	
code_hash	
purpose	enum(registration,password_reset)
metadata	json
expires_at	timestamptz NN
is_used	boolean NN
created_at	timestamptz NN

AuditLog

audit_id	bigint
event_id	uuid NN
event_timestamp	timestamptz NN
table_name	varchar(100) NN
record_id	bigint NN
action	varchar(20) NN
user_id	bigint
request_id	varchar(128)
ip_address	inet
user_agent	text
service_name	varchar(50)
old_data	jsonb
new_data	jsonb
changes	jsonb
checksum	varchar(64)

Store

store_id	bigint
seller_id	bigint NN
store_name	varchar(200) NN
store_description	text
store_address	text
store_email	varchar(100)
store_phone	varchar(20)
total_stars	int
average_rating	decimal(2,1)
total_reviews	int
deleted_at	timestamptz
created_at	timestamptz
updated_at	timestamptz

Seller

seller_id	bigint
user_id	bigint NN
bank_account_reference	varchar(255)
verified	boolean
verified_at	timestamptz
verified_by	bigint
rejected_at	timestamptz
created_at	timestamptz
updated_at	timestamptz

User

user_id	bigint
login_id	varchar(50) NN
password_hash	varchar(255) NN
name	varchar(100) NN
email	varchar(100) NN
avatar_url	varchar(500)
phone_number	varchar(20)
role	enum(buyer,seller,admin) NN
deleted_at	timestamptz
created_at	timestamptz
updated_at	timestamptz

ShippingAddress

address_id	bigint
user_id	bigint NN
recipient_name	varchar(100) NN
phone	varchar(20) NN
city	varchar(50) NN
district	varchar(50)
postal_code	varchar(10)
address_line1	varchar(255) NN
address_line2	varchar(255)
is_default	boolean

OrderItem

order_item_id	bigint
order_id	bigint NN
product_id	bigint
product_snapshot	jsonb NN
quantity	int NN
original_price	decimal(10,2) NN
item_discount	decimal(10,2)
unit_price	decimal(10,2) NN
subtotal	decimal(10,2) NN

Product

product_id	bigint
store_id	bigint NN
product_type_id	bigint NN
product_name	varchar(200) NN
description	text
price	decimal(10,2) NN
sold_count	bigint
total_stars	int
average_rating	decimal(2,1)
total_reviews	int
deleted_at	timestamptz
created_at	timestamptz
updated_at	timestamptz

UserToken

token_id	bigint
user_id	bigint NN
token_hash	varchar(255) NN
expires_at	timestamptz NN
is_revoked	boolean
created_at	timestamptz

OrderDiscount

order_discount_id	bigint
order_id	bigint NN
discount_id	bigint NN
discount_type	enum(seasonal,shipping,special) NN
discount_amount	decimal(10,2) NN
applied_at	timestamptz

Inventory

inventory_id	bigint
product_id	bigint NN
quantity	int NN
reserved	int NN
last_updated	timestamptz NN

ProductImage

image_id	bigint
product_id	bigint NN
image_url	varchar(500) NN
display_order	int NN

CartItem

cart_item_id	bigint
user_id	bigint NN
product_id	bigint NN
quantity	int NN
selected	boolean
added_at	timestamptz
updated_at	timestamptz

CartHistory

cart_history_id	bigint
user_id	bigint NN
cart_snapshot	jsonb NN
item_count	int NN
order_ids	bigint[]
created_at	timestamptz

SpecialDiscount

special_discount_id	bigint
discount_id	bigint NN
store_id	bigint NN
product_type_id	bigint
discount_rate	decimal(5,4)
max_discount_amount	decimal(10,2)

ProductType

product_type_id	bigint
type_code	varchar(50) NN
type_name	varchar(100) NN
parent_type_id	bigint
is_active	boolean

Discount

discount_id	bigint
discount_code	varchar(50) NN
discount_type	enum(seasonal,shipping,special) NN
name	varchar(200) NN
description	text
min_purchase_amount	decimal(10,2) NN
start_datetime	timestamptz NN
end_datetime	timestamptz NN
is_active	boolean
usage_limit	int
usage_count	int
created_by_type	enum(system,seller) NN
created_by_id	bigint
created_at	timestamptz

Order

order_id	bigint
order_number	varchar(50) NN
user_id	bigint NN
store_id	bigint NN
order_status	enum(pending_payment,payment_failed,paid,processing,shipped,delivered,completed,cancelled) NN
subtotal	decimal(10,2) NN
shipping_fee	decimal(10,2) NN
total_discount	decimal(10,2)
total_amount	decimal(10,2) NN
payment_method	varchar(50)
payment_reference	varchar(255)
idempotency_key	varchar(128)
shipping_address_snapshot	jsonb NN
notes	text
created_at	timestamptz
updated_at	timestamptz
paid_at	timestamptz
shipped_at	timestamptz
delivered_at	timestamptz
completed_at	timestamptz
cancelled_at	timestamptz

SeasonalDiscount

seasonal_discount_id	bigint
discount_id	bigint NN
discount_rate	decimal(5,4) NN
max_discount_amount	decimal(10,2)

ShippingDiscount

shipping_discount_id	bigint
discount_id	bigint NN
discount_amount	decimal(10,2) NN

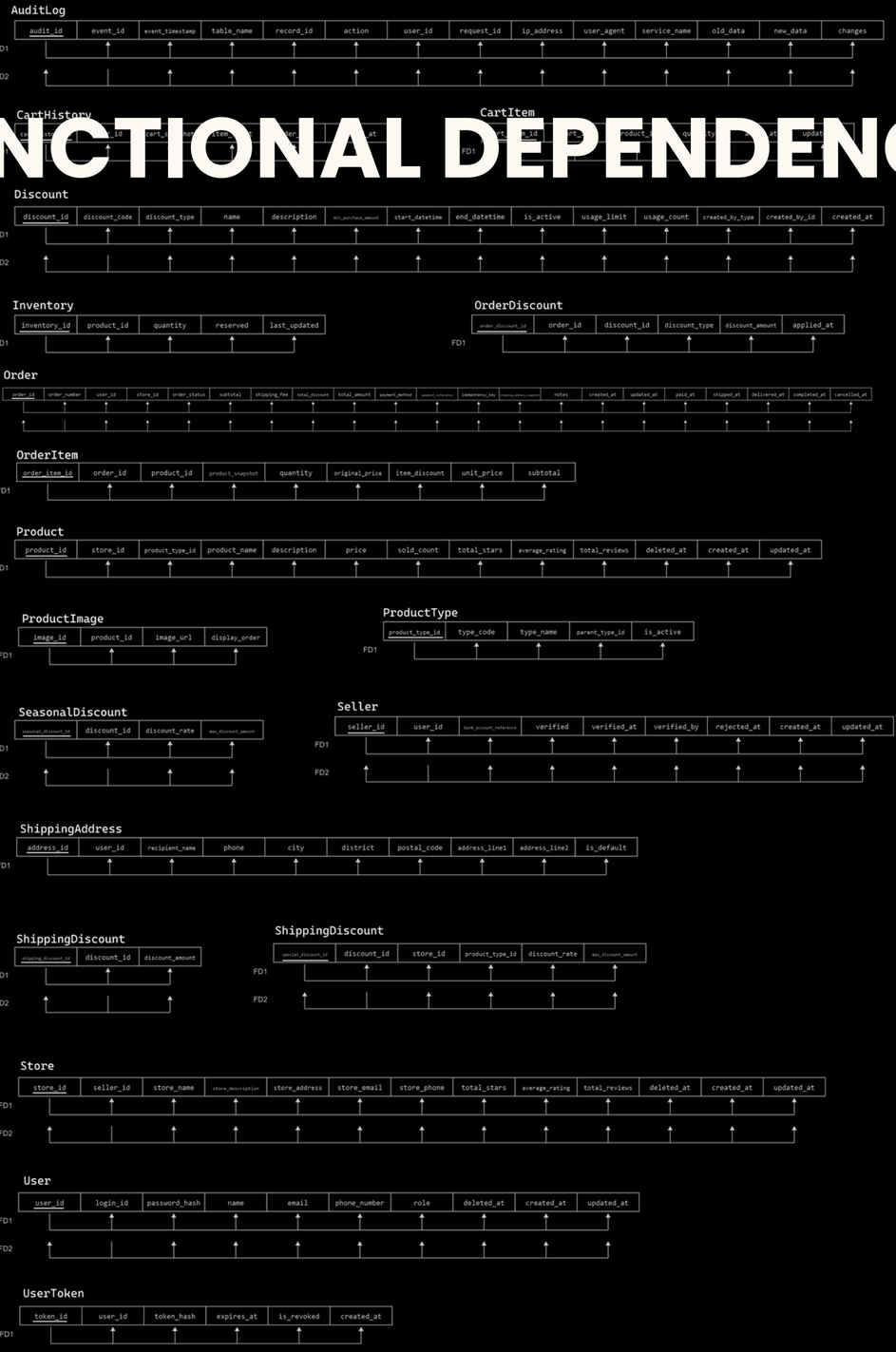
Review

review_id	bigint
product_id	bigint NN
user_id	bigint NN
rating	int NN
comment	text NN
created_at	timestamptz
updated_at	timestamptz

ReviewImage

review_image_id	bigint
review_id	bigint NN
image_url	varchar(500)
public_id	varchar(50)
display_order	int

FUNCTIONAL DEPENDENCY

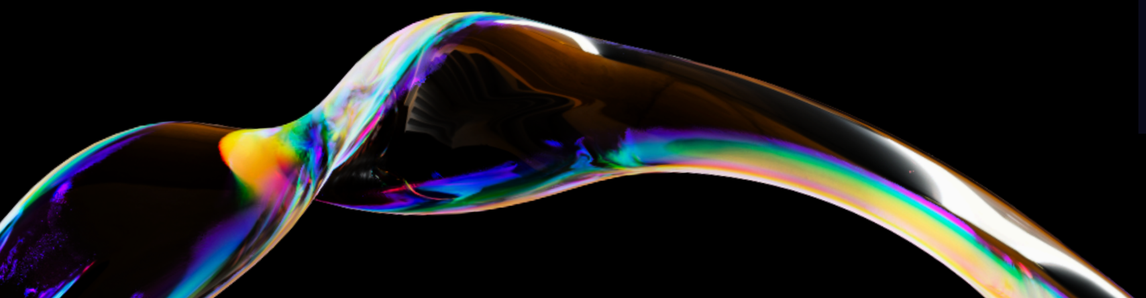


TYPEORM (創建用戶)

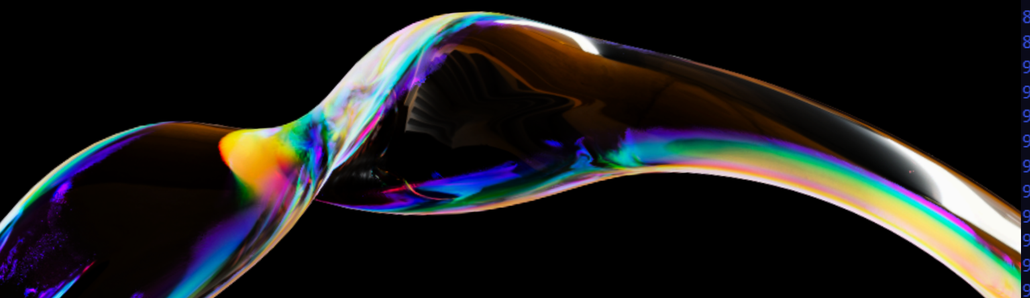
```typescript
export class UsersService {
  constructor(
    @InjectRepository(User)
    private readonly userRepository: Repository<User>,
  ) {}

  async create(createUserDto: CreateUserDto): Promise<User> {
    // Check if loginId or email already exists (excluding soft-deleted users)
    const existingUser = await this.userRepository.findOne({
      where: [
        { loginId: createUserDto.loginId, deletedAt: IsNull() },
        { email: createUserDto.email, deletedAt: IsNull() },
      ],
    });

    //conflict
    if (existingUser) {
      if (existingUser.loginId === createUserDto.loginId) {
        throw new ConflictException('Login ID already exists');
      }
      throw new ConflictException('Email already exists');
    }
    if (!Object.values(UserRole).includes(createUserDto.role as UserRole)) {
      throw new ConflictException('Invalid user role');
    }

    // Hash password
    const saltRounds = 10;
    const passwordHash = await bcrypt.hash(createUserDto.password, saltRounds);

    const user = this.userRepository.create({
      ...createUserDto,
      phoneNumber: formatPhoneNumber(createUserDto.phoneNumber),
      passwordHash,
    });

    return await this.userRepository.save(user);
```

TYPEORM
(取得结账摘要)

```
async getCartSummary(userId: string): Promise<{
  cart: CartItem[];
  totalItems: number;
  totalAmount: number;
  selectedTotalAmount: number;
}> {
  const items = await this.getCart(userId);

  let totalItems = 0;
  let totalAmount = 0;
  let selectedTotalAmount = 0;

  for (const item of items) {
    const price = Number(item.product?.price || 0);
    const itemSubtotal = price * item.quantity;
    totalItems += item.quantity;
    totalAmount += itemSubtotal;

    if (item.selected) selectedTotalAmount += itemSubtotal;
  }

  return {
    cart: items,
    totalItems,
    totalAmount,
    selectedTotalAmount,
  };
}
```



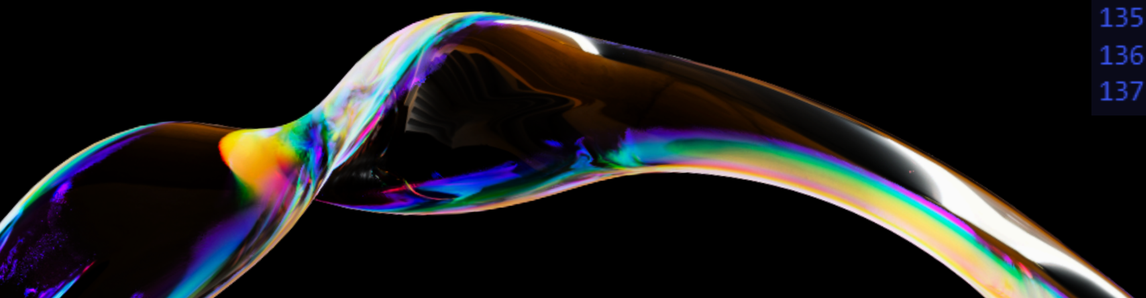
TYPEORM (查詢電商商品列表)



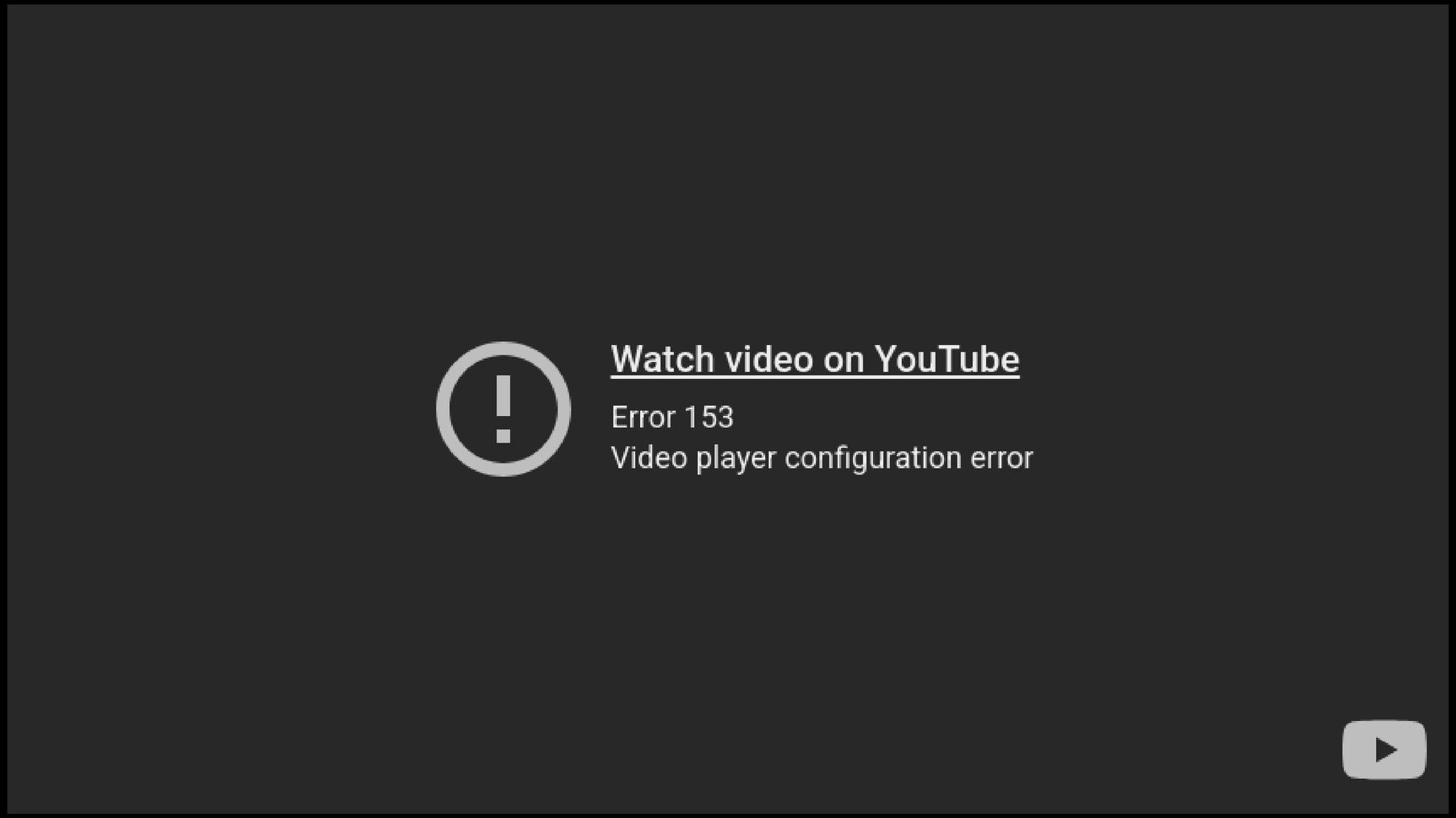
```
45  async findStorefront(
46    queryDto: QueryProductDto,
47  ): Promise<{ data: EnrichedProduct[]; total: number }> {
48    // console.log(queryDto, '=>', queryDto.productTypeId);
49    const page = parseInt(queryDto.page || '1', 10);
50    const limit = parseInt(queryDto.limit || '20', 10);
51    const skip = (page - 1) * limit;
52
53    // Step 1: Query for IDs and Discount Info with Filtering
54    // This query handles filtering, sorting, and pagination
55    const subQueryBuilder =
56      this.productRepository.createQueryBuilder('product');
57
58    // Only join tables needed for filtering/sorting
59    subQueryBuilder
60      .leftJoin(
61        SpecialDiscount,
62        'discount',
63        'discount.storeId = product.storeId AND (discount.productTypeId IS NULL OR discount.productTypeId = product.productTypeId)',
64      )
65      .leftJoin(
66        'discount.discount',
67        'discountInfo',
68        'discountInfo.isActive = true AND discountInfo.startDatetime <= NOW() AND discountInfo.endDatetime >= NOW()',
69      );
70
71    // --- Sorting ---
72    const sortBy = queryDto.sortBy || 'createdAt';
73    const sortOrder =
74      (queryDto.sortOrder?.toUpperCase() as 'ASC' | 'DESC') || 'DESC';
75
76    // Select ID and Max Discount Rate and Sort Column
77    subQueryBuilder
78      .select('product.productId', 'id')
79      .addSelect('MAX(discount.discountRate)', 'maxDiscountRate')
80      .groupBy('product.productId');
81
82    if (sortBy === 'price') {
83      const priceCalc =
84        'MAX(product.price) * (1 - COALESCE(MAX(discount.discountRate), 0))';
85      subQueryBuilder.addSelect(priceCalc, 'sortPrice');
86      subQueryBuilder.addOrderBy('sortPrice', sortOrder);
87    } else {
88      // For strict mode, we should use an aggregate or include in GROUP BY
89      // Since productId is the primary key, product.any_column is functionally dependent on it
90      // but some DB modes still complain. Using MAX() is a safe workaround for non-aggregate columns.
91      subQueryBuilder.addSelect(`MAX(product.${sortBy})`, 'sortVal');
92      subQueryBuilder.addOrderBy('sortVal', sortOrder);
93    }
94
95    // --- Filters ---
96    subQueryBuilder.andWhere('product.isActive = :isActive', {
97      isActive: true,
98    });
99
```

TYPEORM
(查詢評論統計)

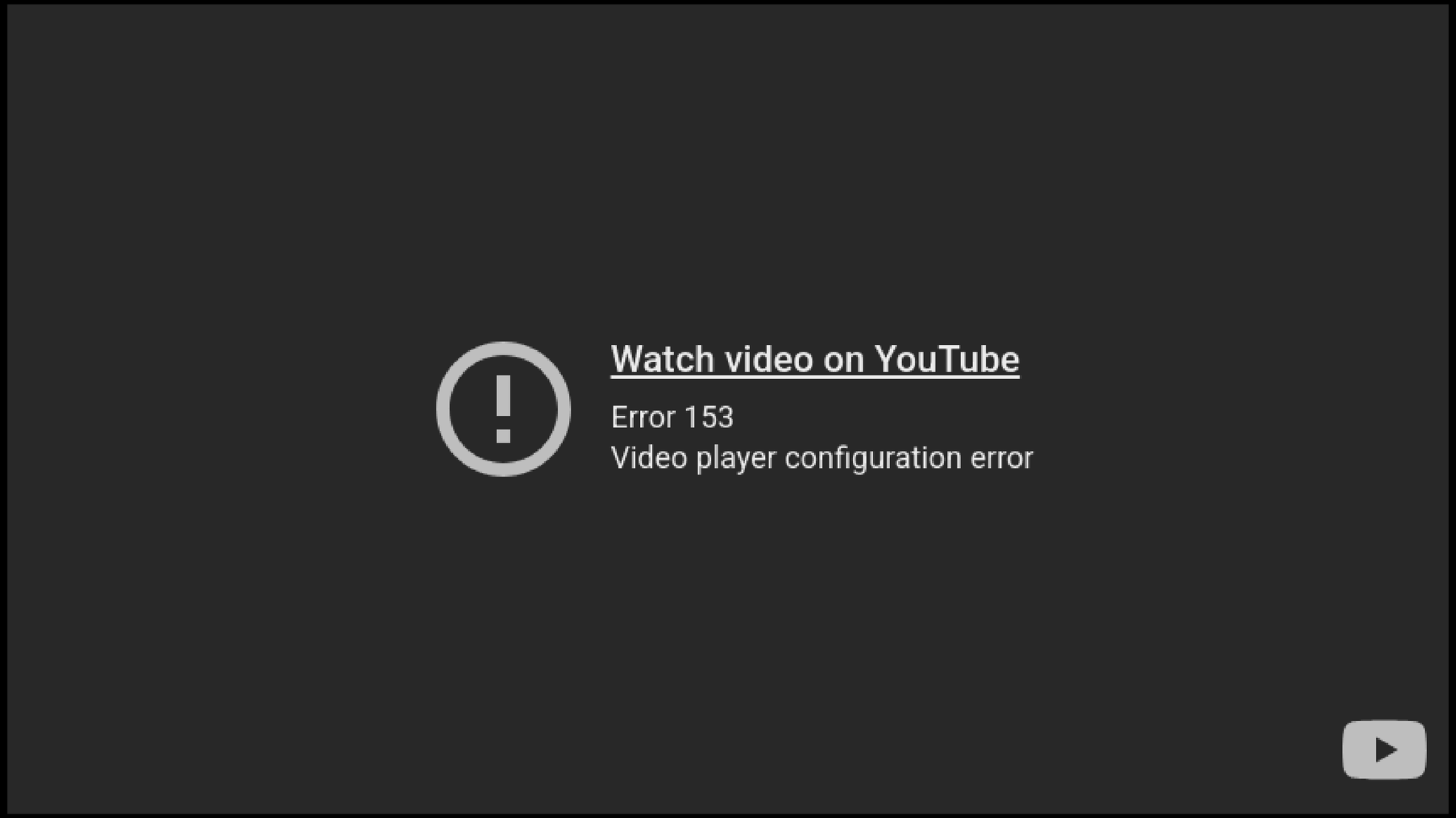
```
110   async getStatistics(productId: string) {
111     const stats = await this.reviewRepository
112       .createQueryBuilder('review')
113       .select('review.rating', 'rating')
114       .addSelect('COUNT(*)', 'count')
115       .where('review.productId = :productId', { productId })
116       .groupBy('review.rating')
117       .getRawMany();
118
119     const distribution = { 1: 0, 2: 0, 3: 0, 4: 0, 5: 0 };
120     let totalScore = 0;
121     let totalCount = 0;
122
123     stats.forEach((s) => {
124       const r = parseInt(s.rating);
125       const c = parseInt(s.count);
126       distribution[r] = c;
127       totalScore += r * c;
128       totalCount += c;
129     });
130
131     return {
132       average:
133         totalCount > 0 ? Math.round((totalScore / totalCount) * 10) / 10 : 0,
134       total: totalCount,
135       distribution,
136     };
137   }
```



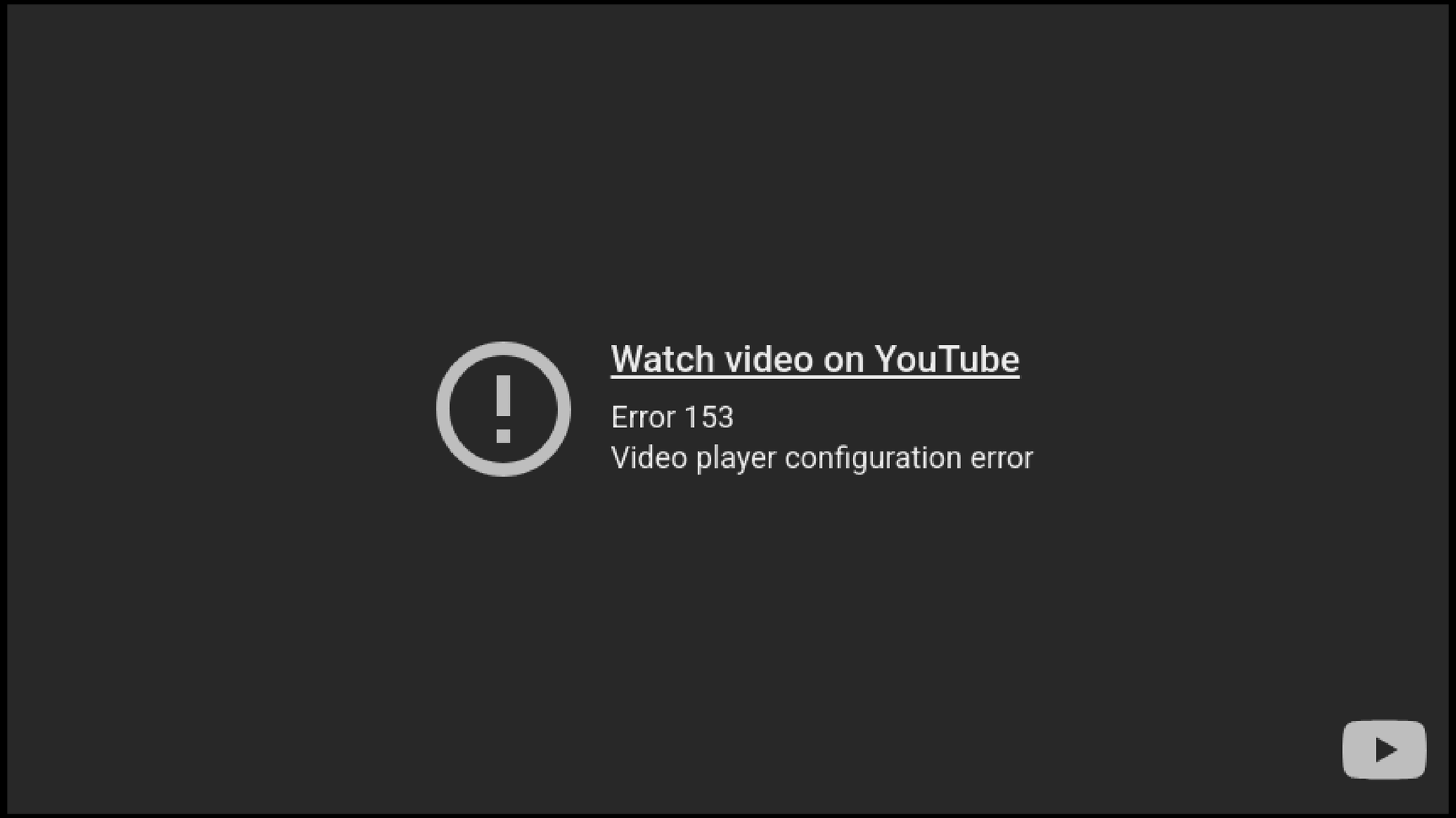
VISITOR DEMO



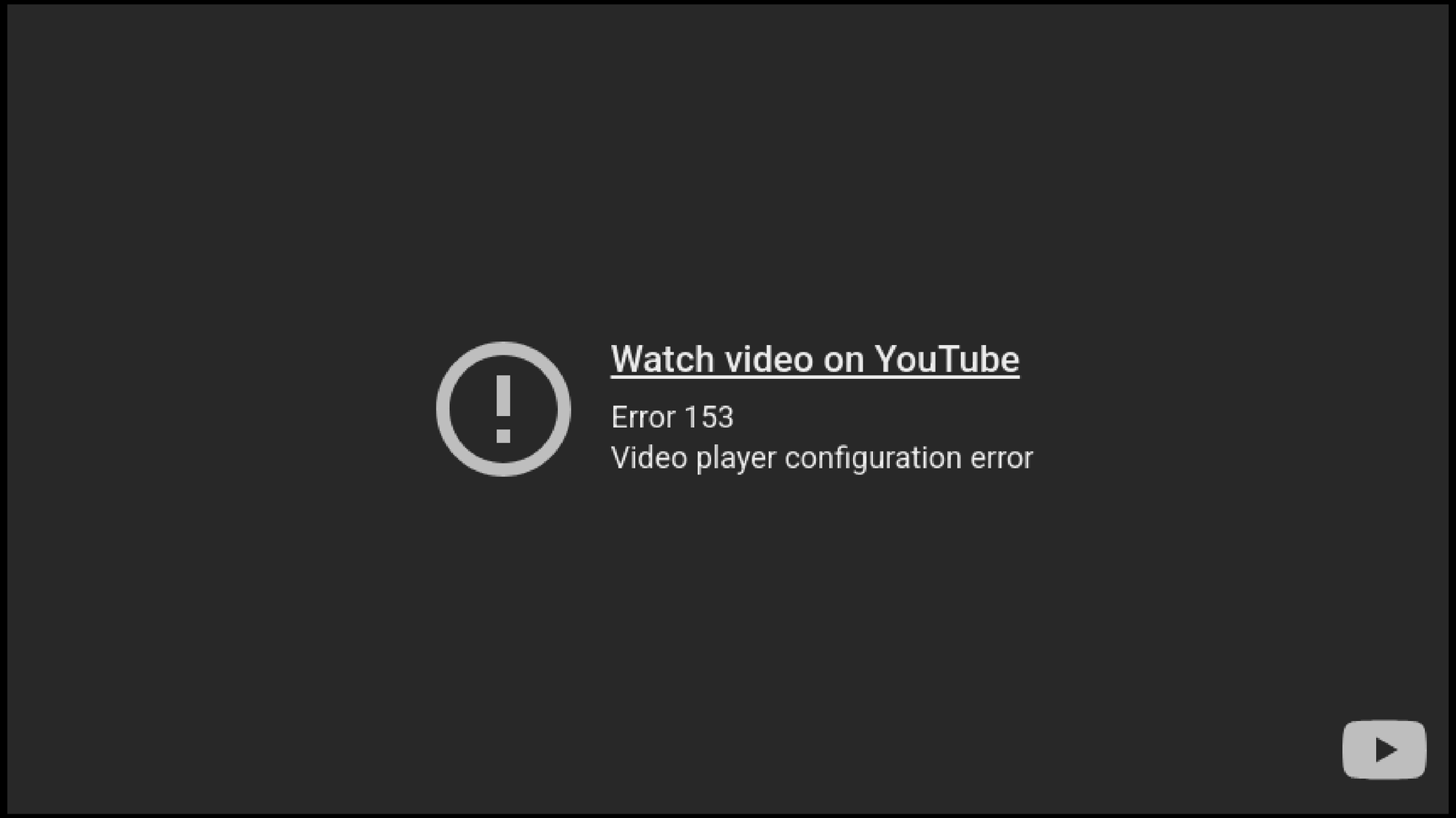
BUYER DEMO


Watch video on YouTube
Error 153
Video player configuration error

SELLER DEMO


Watch video on YouTube
Error 153
Video player configuration error

ADMIN DEMO


Watch video on YouTube
Error 153
Video player configuration error

CONTRIBUTIONS

FRONTEND

Yujun

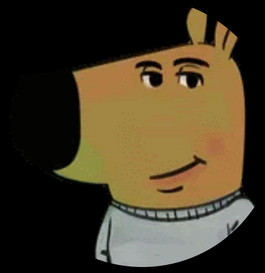
胡育鈞　12.5%

FULLSTACK

Julian

鄭錦鑫　35%

BACKEND



Chill Kai

葉凱成　20%

Chuni

陳駿逸　20%

Tata

高翎竣　12.5%

Group 4 In Database System Lecture

THANK YOU

for your time and attention

https://aps.ntut.edu.tw/course/tw/Curr.jsp?format=-2&code=5902206

hyujun940529@gmail.com

106344 No. 1, Sec. 3,
Zhongxiao E. Rd., Da'an Dist., Taipei City