HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Final Report

Fashion E-commerce test Website

Final Database Presentation

Supervisor: Nguyễn Hồng Phương

Group Members:
Bùi Doãn Khang 20235950
Nguyễn Tuấn Đức 202359
Bùi Quang Minh 202359

ONE LOVE. ONE FUTURE.

Table Contents

HUST

hust.edu.vn fb.com/dhbkhn

Project Context & Motivation

I. Project Context & Motivation

- **Market Trend:** The rapid shift from brick-and-mortar retail to digital-first fashion ecosystems.
- **Legacy Challenges:**
    - **Data Fragmentation:** Using Excel and paper leads to many mistakes because too slow to update data
    - **Inventory Desynchronization:** Website and Warehouse numbers do not match (discrepancy)
    - **Scalability Bottlenecks:** Inability to handle high-traffic seasonal sales.
- **Our Mission:** Building a high-performance "Operational Backbone" using Microsoft SQL Server to ensure 100% data integrity.

I. Project Context & Motivation

Core Technical Challenges:

- **Concurrency Control:** Preventing "Overselling" during simultaneous purchases via ACID transactions as well as triggers.
- **Search Performance:** Optimizing discovery for 50,000+ SKUs using indexing strategies.
- **Logic Coupling:** Decoupling business rules from the frontend and centralizing them within **T-SQL Stored Procedures** for security and consistency.

HUST

hust.edu.vn fb.com/dhbkhn

System Architecture & Database Design

II. System Architecture & Database Design

Technology stack:

Database Level (The System Core)

- **MS SQL Server 2022:** Industrial-grade engine for robust transaction management and high security.
- **T-SQL Stored Procedures:** All critical business logic (Orders, Inventory, Vouchers) is encapsulated in the database to ensure **execution speed** and **data consistency**.

Application Level (The Logic Bridge)

- **Node.js & Express:** A non-blocking, event-driven runtime environment designed to handle high-concurrency e-commerce traffic.
- **EJS (Embedded JS):** Server-side rendering for dynamic and SEO-friendly content delivery.

Frontend Level (The User Interface)

- **Bootstrap 5:** Ensures a professional, **mobile-first**, and fully responsive design across all devices.

II. System Architecture & Database Design

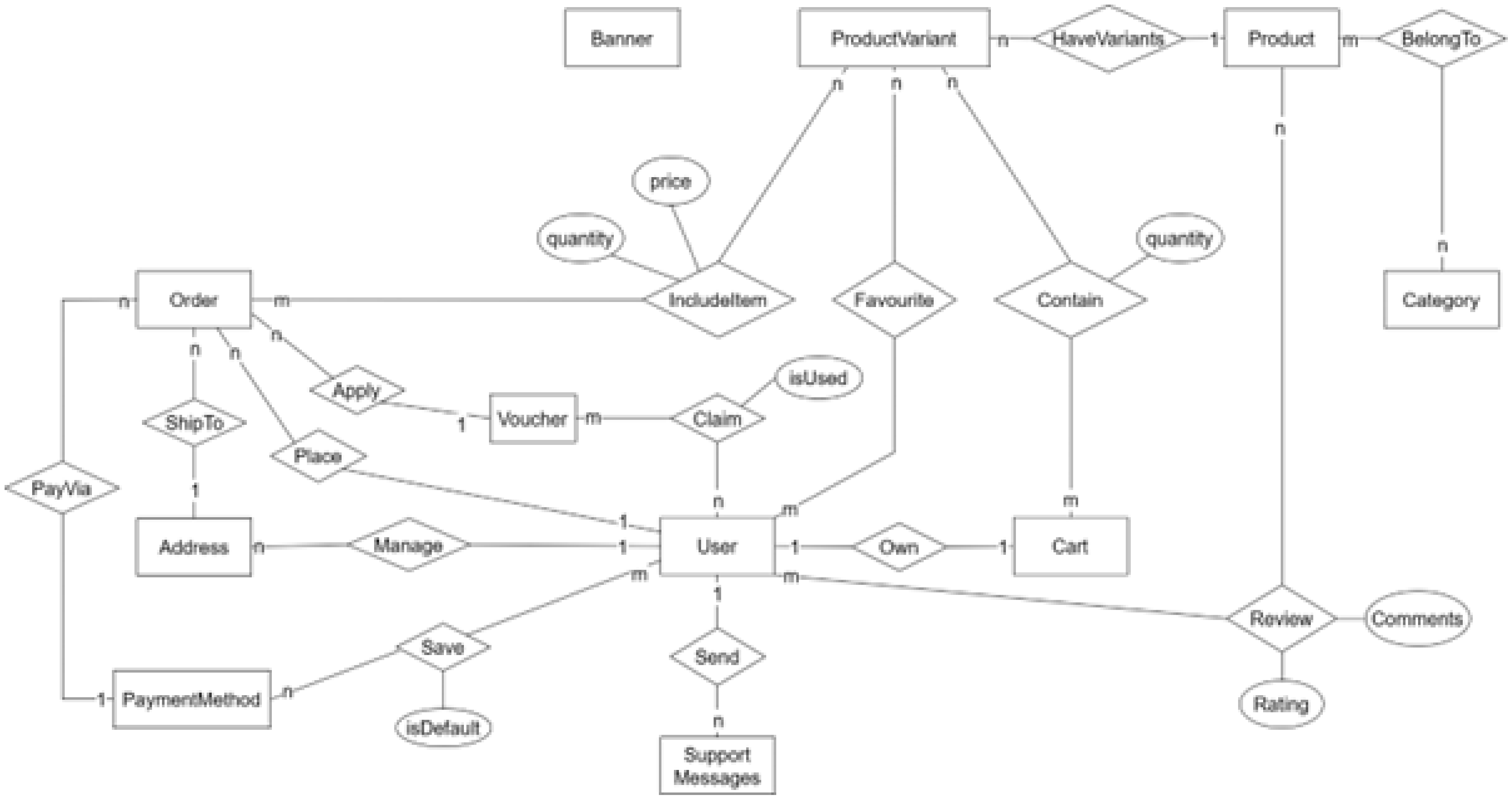
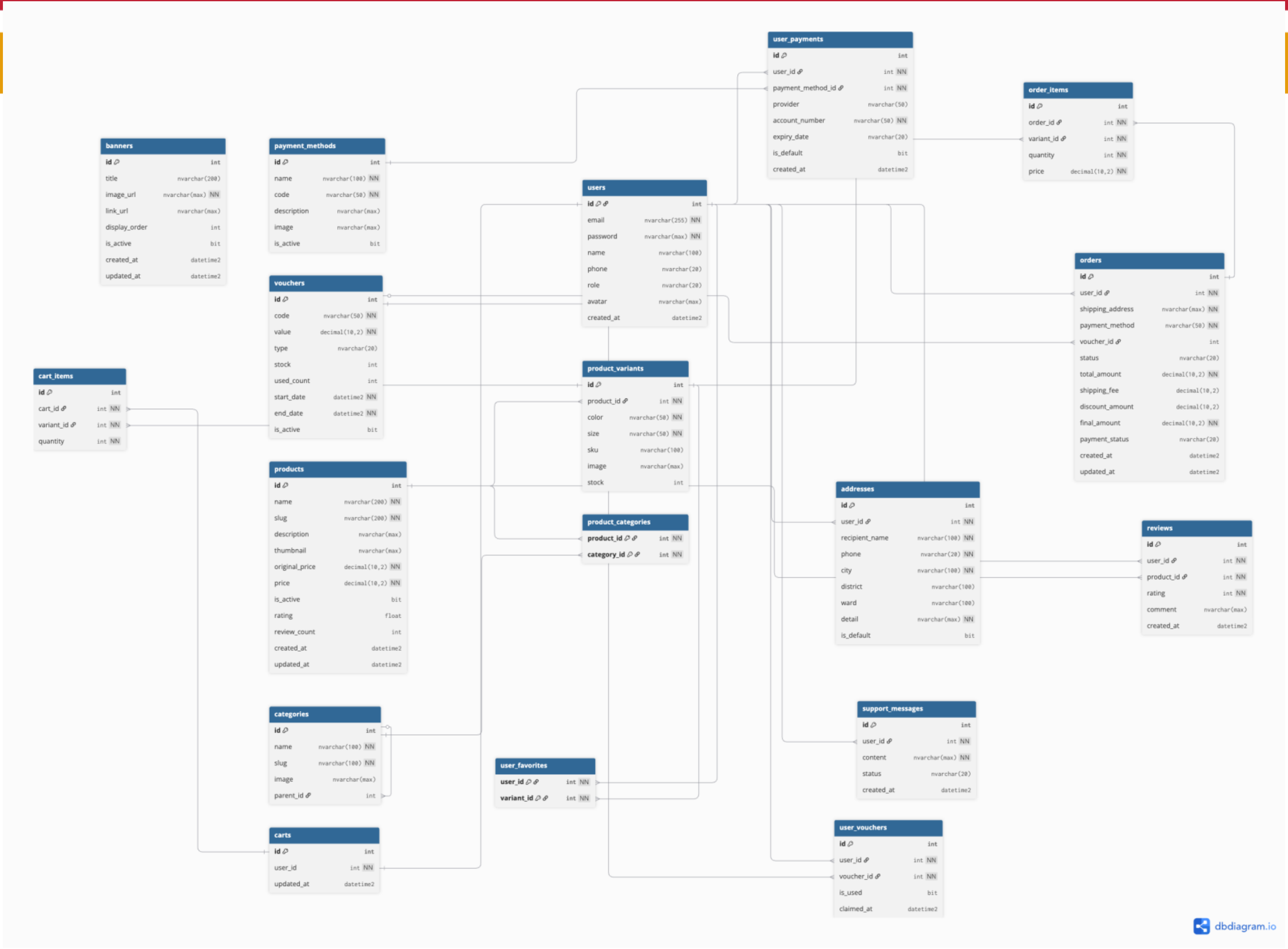


figure 2.1: ERD diagram

II. System Architecture & Database Design



user_payments: id int; user_id int NN; payment_method_id int NN; provider nvarchar(50); account_number nvarchar(50) NN; expiry_date nvarchar(20); is_default bit; created_at datetime2
order_items: id int; order_id int NN; variant_id int NN; quantity int NN; price decimal(10,2) NN
banners: id int; title nvarchar(200); image_url nvarchar(max) NN; link_url nvarchar(max); display_order int; is_active bit; created_at datetime2; updated_at datetime2
payment_methods: id int; name nvarchar(100) NN; code nvarchar(50) NN; description nvarchar(max); image nvarchar(max); is_active bit
users: id int; email nvarchar(255) NN; password nvarchar(max) NN; name nvarchar(100); phone nvarchar(20); role nvarchar(20); avatar nvarchar(max); created_at datetime2
orders: id int; user_id int NN; shipping_address nvarchar(max) NN; payment_method nvarchar(50) NN; voucher_id int; status nvarchar(20); total_amount decimal(10,2) NN; shipping_fee decimal(10,2); discount_amount decimal(10,2); final_amount decimal(10,2) NN; payment_status nvarchar(20); created_at datetime2; updated_at datetime2
vouchers: id int; code nvarchar(50) NN; value decimal(10,2) NN; type nvarchar(20); stock int; used_count int; start_date datetime2 NN; end_date datetime2 NN; is_active bit
cart_items: id int; cart_id int NN; variant_id int NN; quantity int NN
product_variants: id int; product_id int NN; color nvarchar(50) NN; size nvarchar(50) NN; sku nvarchar(100); image nvarchar(max); stock int
products: id int; name nvarchar(200) NN; slug nvarchar(200) NN; description nvarchar(max); thumbnail nvarchar(max); original_price decimal(10,2) NN; price decimal(10,2) NN; is_active bit; rating float; review_count int; created_at datetime2; updated_at datetime2
addresses: id int; user_id int NN; recipient_name nvarchar(100) NN; phone nvarchar(20) NN; city nvarchar(100) NN; district nvarchar(100); ward nvarchar(100); detail nvarchar(max) NN; is_default bit
product_categories: product_id int NN; category_id int NN
reviews: id int; user_id int NN; product_id int NN; rating int NN; comment nvarchar(max); created_at datetime2
categories: id int; name nvarchar(100) NN; slug nvarchar(100) NN; image nvarchar(max); parent_id int
support_messages: id int; user_id int NN; content nvarchar(max) NN; status nvarchar(20); created_at datetime2
user_favorites: user_id int NN; variant_id int NN
carts: id int; user_id int NN; updated_at datetime2
user_vouchers: id int; user_id int NN; voucher_id int NN; is_used bit; claimed_at datetime2
dbdiagram.io

figure 2.2: Relational Model

II. System Architecture & Database Design

figure 2.2: Relational Model

HUST

hust.edu.vn fb.com/dhbkhn

Functional implementation map

- Customer module
- Admin module

III. Functional implementation map (Customer module)

Authentication & Profile:

Procedure Name	**Description & Implementation Note**
`register_user`	Creates new account with CHECK role constraints. Returns new User ID.
`login_user`	Validates credentials (email/password) and returns user profile + id.
`update_profile`	Updates personal info (Avatar, Phone, Password).
`get_my_addresses`	Returns list of all saved shipping addresses for the logged-in user.
`add_address`	Adds a new address. If `is_default` is true, automatically updates other addresses to false.
`update_address`	Updates existing address details.
`delete_address`	**Soft/Hard Delete**: Removes an address from the user's book.

III. Functional implementation map (Customer module)

Product Discovery & Browsing:

Procedure Name	Description & Implementation Note
`browse_products`	**Complex Search**: Handles Keywords, Category Slug, Attributes (Color/Size), Price Range. Implements Dynamic Sorting and Pagination.
`get_product_details`	**JSON Return**: Returns nested JSON object containing Product Info, Variants List, and Latest Reviews in a single query.
`get_trending_products`	Returns top-selling items based on last 30 days.

III. Functional implementation map (Customer module)

Shopping Cart System:

Procedure Name	Description & Implementation Note
`cart_add_item`	**Smart Upsert**: Uses MERGE statement. Validates stock limit (`Current + New <= Stock`) before adding.
`cart_update_item_quantity`	Modifies quantity. Deletes item if New Quantity <= 0.
`cart_remove_item`	Removes item from cart permanently.
`cart_view_details`	Lists items with calculated subtotals and real-time stock availability check.

HUST
ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

III. Functional implementation map (Customer module)

Checkout & Orders:

Procedure Name	Description & Implementation Note
`checkout`	**ACID Transaction**: Atomic operation handling Stock Deduction, Voucher Application, Address Snapshotting, and Order Creation. Uses `BEGIN TRANSACTION`.
`collect_voucher`	Validates code validity/usage limits and adds to user's wallet.
`view_my_vouchers`	Lists available vouchers with status (Ready, Expired, Out of Stock).
`view_order_history`	Lists all orders with status (Pending, Shipping, etc.).
`cancel_order`	**Restock Logic**: Allows user to cancel 'PENDING' orders. Automatically restores Product Stock and Vouchers. Uses UPDLOCK.

III. Functional implementation map (Customer module)

Engagement & Support:

Procedure Name	Description & Implementation Note
`view_wishlist`	Displays all products currently saved in the user's favorites list.
`add_to_wishlist`	Adds product variant to `user_favorites`.
`remove_from_wishlist`	Removes a product from the user's favorites list.
`submit_product_review`	Allows verified purchasers to rate (1-5) and comment. Updates aggregate Product Rating.
`send_support_message`	Creates a new support ticket in `support_messages` table.

III. Functional implementation map (Admin module)

Product Catalog Management:

Procedure Name	Description & Implementation Note
`create_product`	Inserts base product. Handles category linking via JSON array input (`OPENJSON`).
`upsert_variant`	**Upsert Logic**: Adds new variant (Size/Color) or updates existing one using `MERGE`.
`update_product`	Updates general info (Name, Price, Status).
`delete_product`	Soft-delete: Sets `is_active = 0`.
`delete_variant`	Hard-delete: Only allowed if variant has never been sold.

III. Functional implementation map (Admin module)

Marketing & Promotions:

Procedure Name	Description & Implementation Note
`upsert_voucher`	Creates/Updates discounts. Logic includes Quantity, Date Range, Type (Fixed/Percent).
`upsert_banner`	Manages Homepage Banners (Image URL, Link, Display Order).
`delete_voucher`	Deactivates voucher if used, or permanently deletes if unused.
`delete_banner`	Removes banner from the system.

III. Functional implementation map (Admin module)

Order Operations:

Procedure Name	Description & Implementation Note
`view_orders`	Filters orders by Status and Date Range.
`update_order_status`	**Workflow**: Updates status (e.g., Shipping -> Completed). Automated Restocking on 'RETURNED' status.

III. Functional implementation map (Admin module)

Reports & Analytics:

Procedure Name	Description & Implementation Note
`report_revenue_by_date`	Aggregates daily revenue for `COMPLETED` orders.
`report_best_sellers`	Returns top products by quantity sold and total revenue generated.
`report_revenue_by_category`	Insights into which product categories are driving sales.

HUST

hust.edu.vn fb.com/dhbkhn

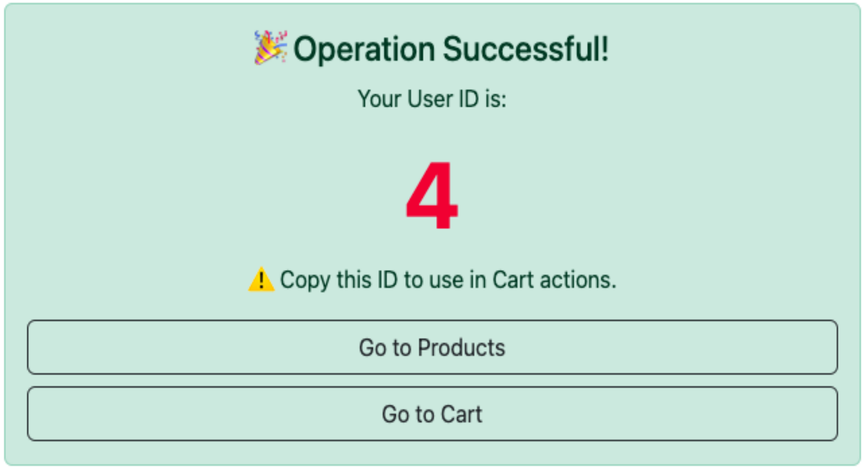
Scenario-based Testing

IV. Scenario-based Testing

- **Objective:** Validate end-to-end functionality via a "New User Journey."
- **Scope:** Account Creation -> Secure Authentication -> Profile Setup -> Transactional Shopping -> Order Management.

Test Data Setup:

- **New Email:** flow_test_@gmail.com
- **Password:** 123456
- **Name:** Test User
- **Phone:** 0988888888



🎉 Operation Successful!

Your User ID is:

4

⚠️ Copy this ID to use in Cart actions.

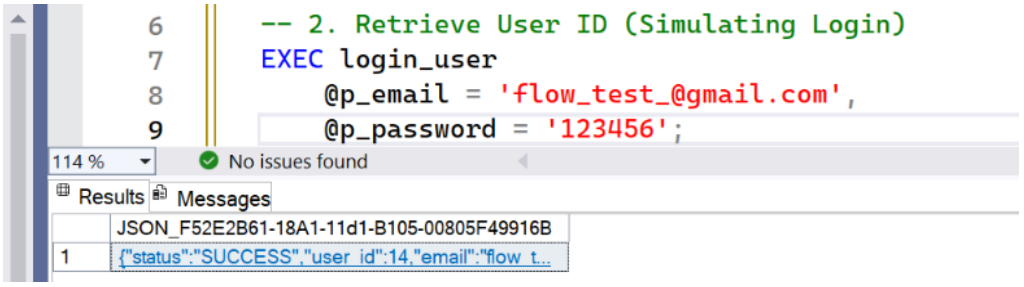
Step 1: User registration

```
-- 1. Register
-- (Using generated email to avoid conflict if run multiple times)
DECLARE @Email NVARCHAR(50) = 'flow_test_@gmail.com';
EXEC register_user @Email, '123456', 'Test User', '0988888888';
```

IV. Scenario-based Testing

Step 2: Log in

- **Action:** The user attempts to log in with the credentials created in Step 1.
- **Objective:** Verify login_user checks credentials correctly and returns a success status.

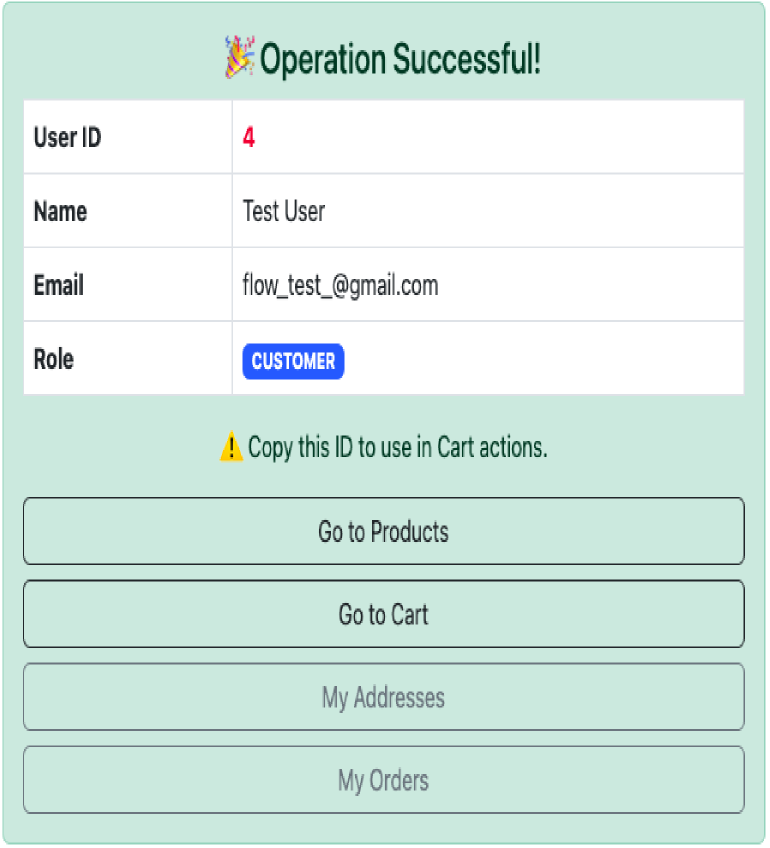


```
-- 2. Retrieve User ID (Simulating Login)
EXEC login_user
    @p_email = 'flow_test_@gmail.com',
    @p_password = '123456';
```

114 % No issues found

Results Messages

```
   JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1  {"status":"SUCCESS","user_id":14,"email":"flow_t...
```



Operation Successful!

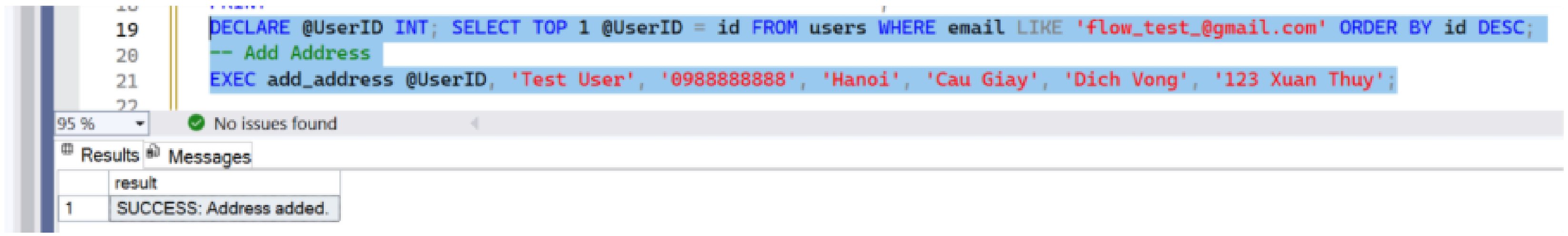
User ID	4
Name	Test User
Email	flow_test_@gmail.com
Role	CUSTOMER

Copy this ID to use in Cart actions.

IV. Scenario-based Testing

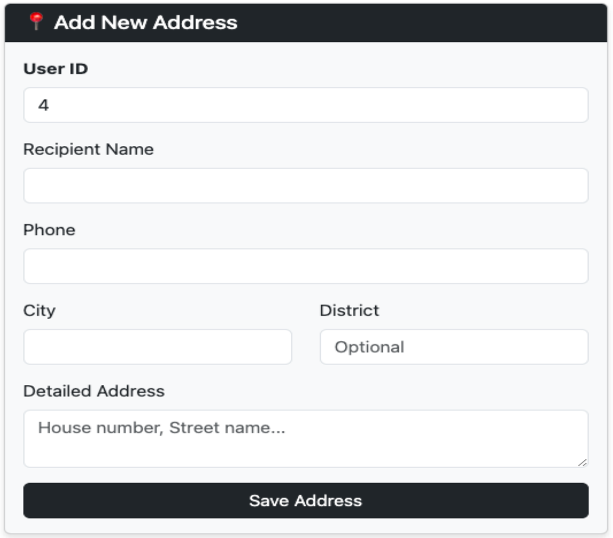
Step 3: Setup Shipping Address

- **Action:** Before buying, the user adds a shipping address to their profile.
- **Objective:** Verify add_address.



```
DECLARE @UserID INT; SELECT TOP 1 @UserID = id FROM users WHERE email LIKE 'flow_test_@gmail.com' ORDER BY id DESC;
-- Add Address
EXEC add_address @UserID, 'Test User', '0988888888', 'Hanoi', 'Cau Giay', 'Dich Vong', '123 Xuan Thuy';
```

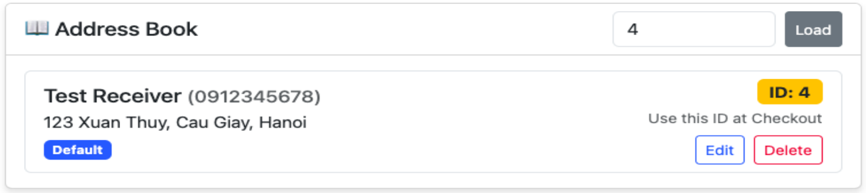
```
95 %    No issues found
Results  Messages
   result
1  SUCCESS: Address added.
```



Add New Address

User ID	4
Recipient Name	
Phone	
City	
District	Optional
Detailed Address	House number, Street name...

Save Address



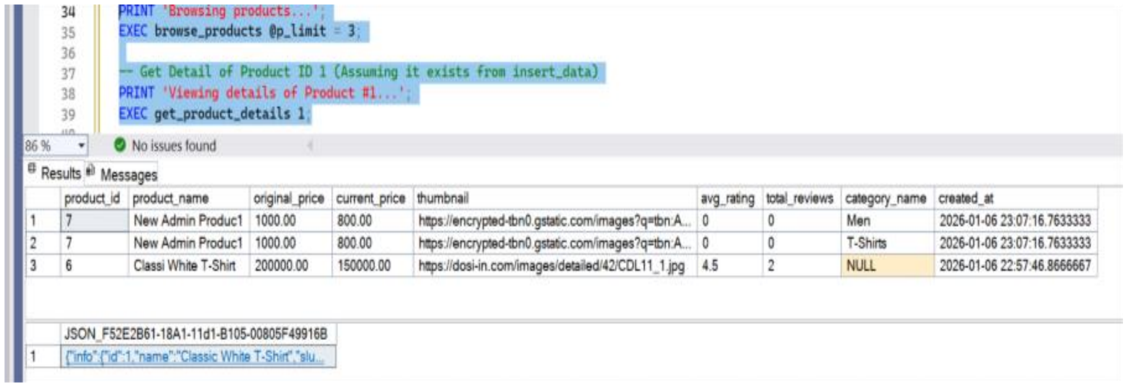
Address Book 4 Load

Test Receiver (0912345678) ID: 4
123 Xuan Thuy, Cau Giay, Hanoi Use this ID at Checkout
Default Edit Delete

IV. Scenario-based Testing

Step 4: Product Discovery

- **Action:** The user searches for 3 TOP items
- **Objective:** Verify browse_products returns relevant results.



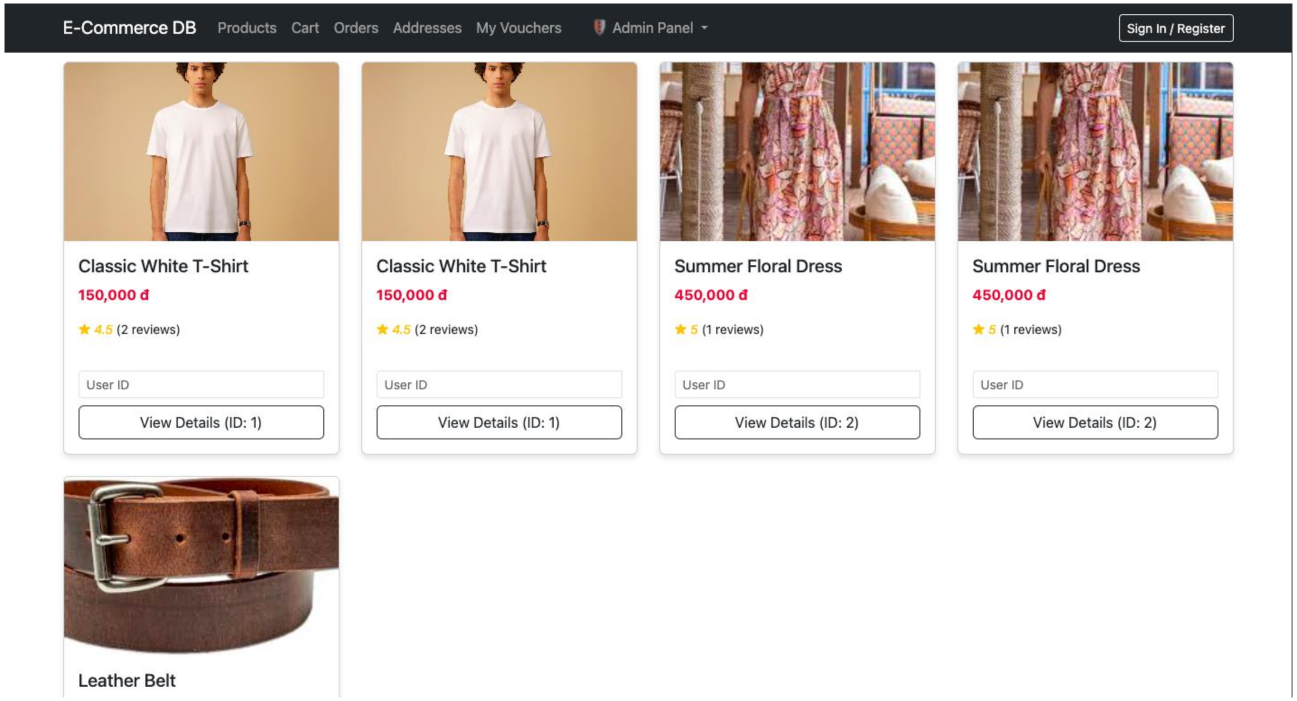
```
PRINT 'Browsing products...';
EXEC browse_products @p_limit = 3;

-- Get Detail of Product ID 1 (Assuming it exists from insert_data)
PRINT 'Viewing details of Product #1...';
EXEC get_product_details 1;
```

86 % No issues found

Results Messages

	product_id	product_name	original_price	current_price	thumbnail	avg_rating	total_reviews	category_name	created_at
1	7	New Admin Produc1	1000.00	800.00	https://encrypted-tbn0.gstatic.com/images?q=tbn:A...	0	0	Men	2026-01-06 23:07:16.7633333
2	7	New Admin Produc1	1000.00	800.00	https://encrypted-tbn0.gstatic.com/images?q=tbn:A...	0	0	T-Shirts	2026-01-06 23:07:16.7633333
3	6	Classi White T-Shirt	200000.00	150000.00	https://dosi-in.com/images/detailed/42/CDL11_1.jpg	4.5	2	NULL	2026-01-06 22:57:46.8666667

	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	{"info":{"id":1,"name":"Classic White T-Shirt","slu...



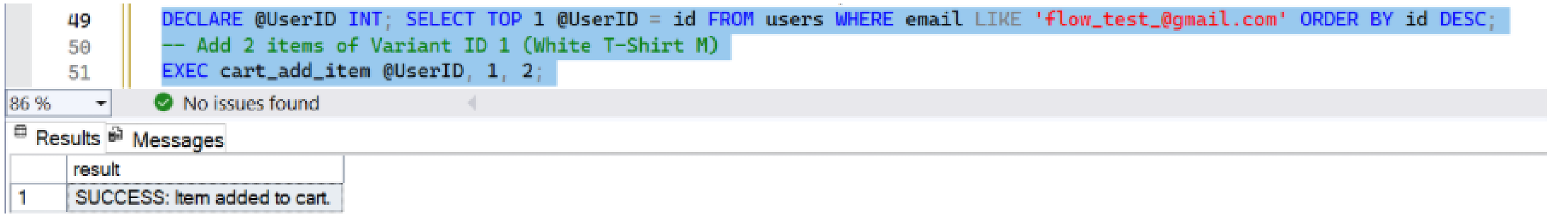
Classic White T-Shirt	Classic White T-Shirt	Summer Floral Dress	Summer Floral Dress
150,000 đ	150,000 đ	450,000 đ	450,000 đ
★ 4.5 (2 reviews)	★ 4.5 (2 reviews)	★ 5 (1 reviews)	★ 5 (1 reviews)
User ID	User ID	User ID	User ID
View Details (ID: 1)	View Details (ID: 1)	View Details (ID: 2)	View Details (ID: 2)

Leather Belt

IV. Scenario-based Testing

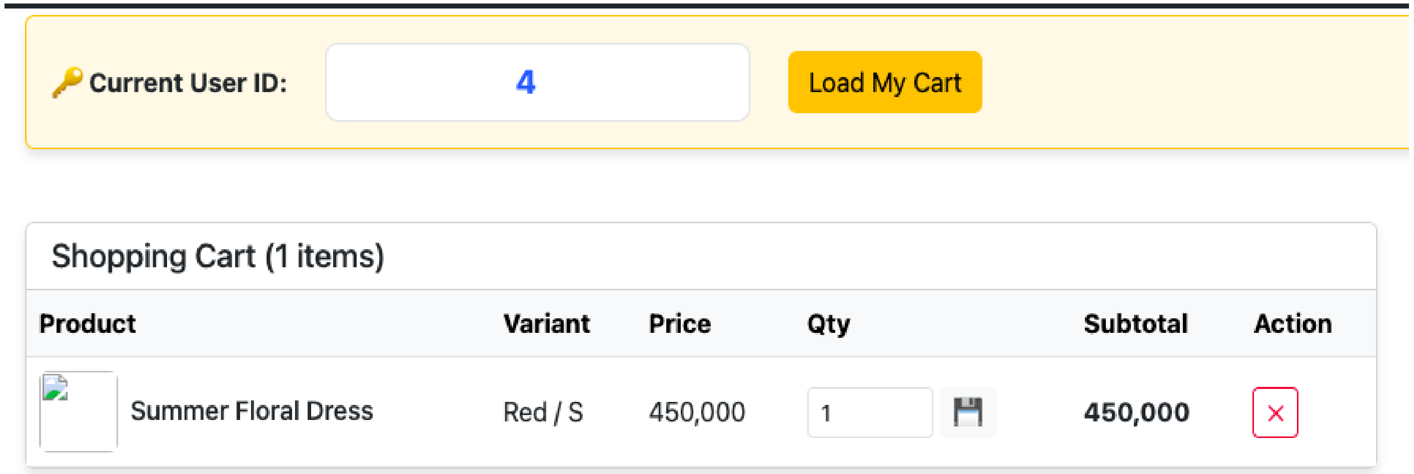
Step 5: Add to Cart

- **Action:** The user adds items with the first variant and quantity is 2
- **Objective:** Verify cart_add_item logic.



```
DECLARE @UserID INT; SELECT TOP 1 @UserID = id FROM users WHERE email LIKE 'flow_test_@gmail.com' ORDER BY id DESC;
-- Add 2 items of Variant ID 1 (White T-Shirt M)
EXEC cart_add_item @UserID, 1, 2;
```

```
86 %    No issues found
Results  Messages
   result
1  SUCCESS: Item added to cart.
```



Current User ID: 4

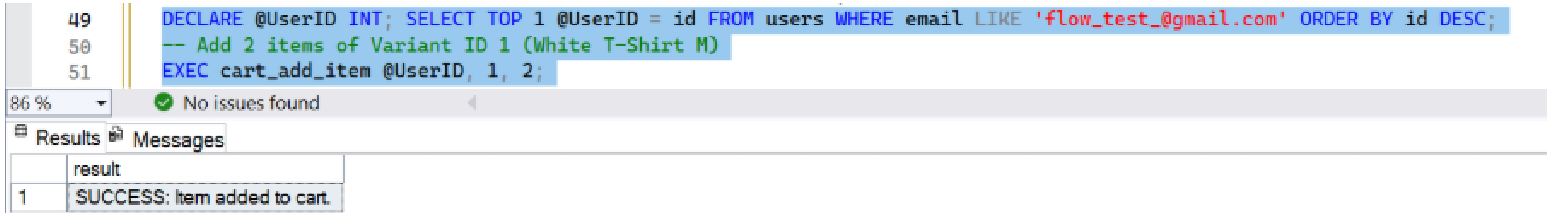
Shopping Cart (1 items)

Product	Variant	Price	Qty	Subtotal	Action
Summer Floral Dress	Red / S	450,000	1	450,000	×

IV. Scenario-based Testing

Step 6: Checkout Transaction

- **Action:** The user places the order using the address created in Step 3.
- **Objective:** Verify checkout creates the order, deducts stock, and clears the cart.

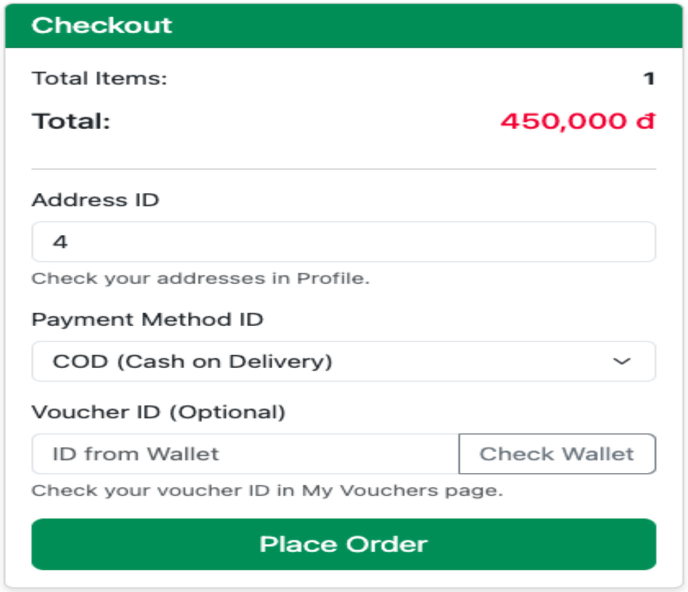


```
DECLARE @UserID INT; SELECT TOP 1 @UserID = id FROM users WHERE email LIKE 'flow_test_@gmail.com' ORDER BY id DESC;
-- Add 2 items of Variant ID 1 (White T-Shirt M)
EXEC cart_add_item @UserID, 1, 2;
```

86 % No issues found

Results Messages

	result
1	SUCCESS: Item added to cart.



Checkout

Total Items: 1

Total: 450,000 đ

Address ID: 4

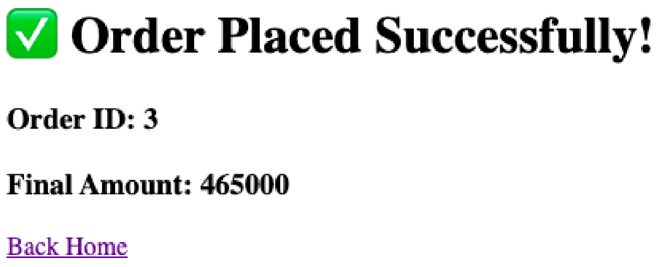
Check your addresses in Profile.

Payment Method ID: COD (Cash on Delivery)

Voucher ID (Optional): ID from Wallet | Check Wallet

Check your voucher ID in My Vouchers page.

Place Order



✅ **Order Placed Successfully!**

Order ID: 3

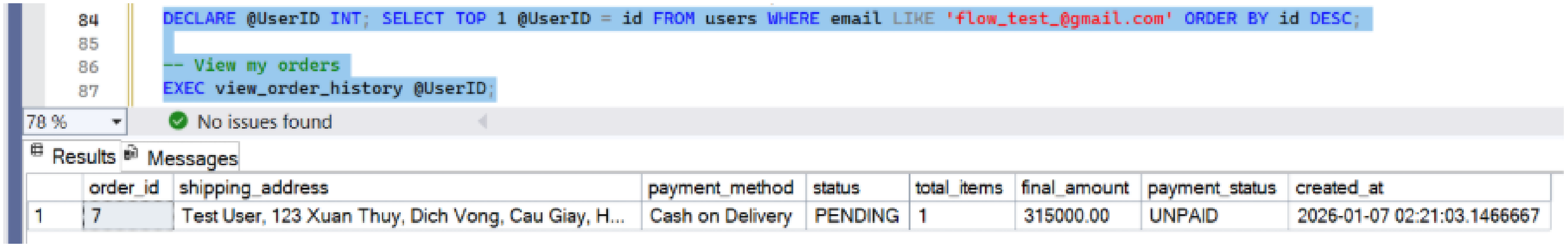
Final Amount: 465000

Back Home

IV. Scenario-based Testing

Step 7: Order Tracking

- **Action:** The user checks their order history to confirm the purchase.
- **Objective:** Verify view_order_history.

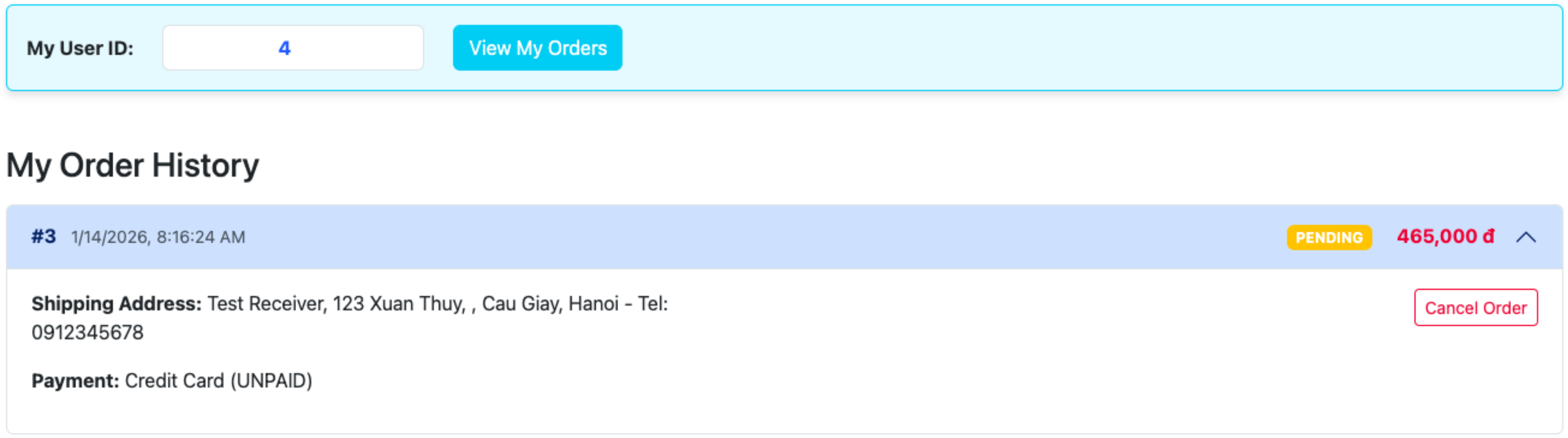


```
DECLARE @UserID INT; SELECT TOP 1 @UserID = id FROM users WHERE email LIKE 'flow_test_@gmail.com' ORDER BY id DESC;

-- View my orders
EXEC view_order_history @UserID;
```

78 % No issues found

Results Messages

	order_id	shipping_address	payment_method	status	total_items	final_amount	payment_status	created_at
1	7	Test User, 123 Xuan Thuy, Dich Vong, Cau Giay, H...	Cash on Delivery	PENDING	1	315000.00	UNPAID	2026-01-07 02:21:03.1466667



My User ID: 4 View My Orders

My Order History

#3 1/14/2026, 8:16:24 AM PENDING 465,000 đ

Shipping Address: Test Receiver, 123 Xuan Thuy, , Cau Giay, Hanoi - Tel: 0912345678

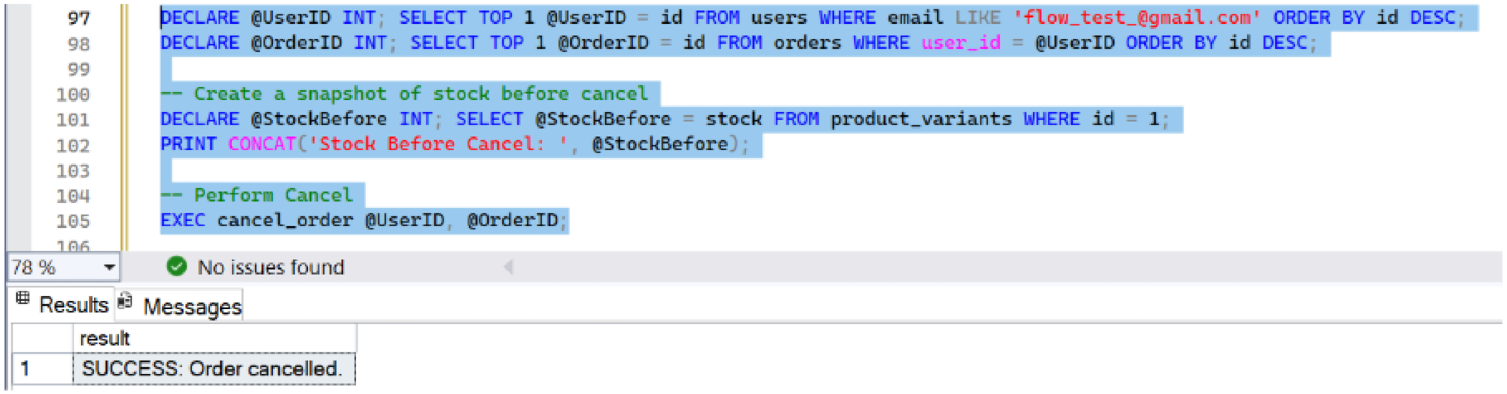
Payment: Credit Card (UNPAID)

Cancel Order

IV. Scenario-based Testing

Step 8: Order Cancellation

- **Action:** The user decides to cancel the order immediately.
- **Objective:** Verify cancel_order restores stock and updates status.

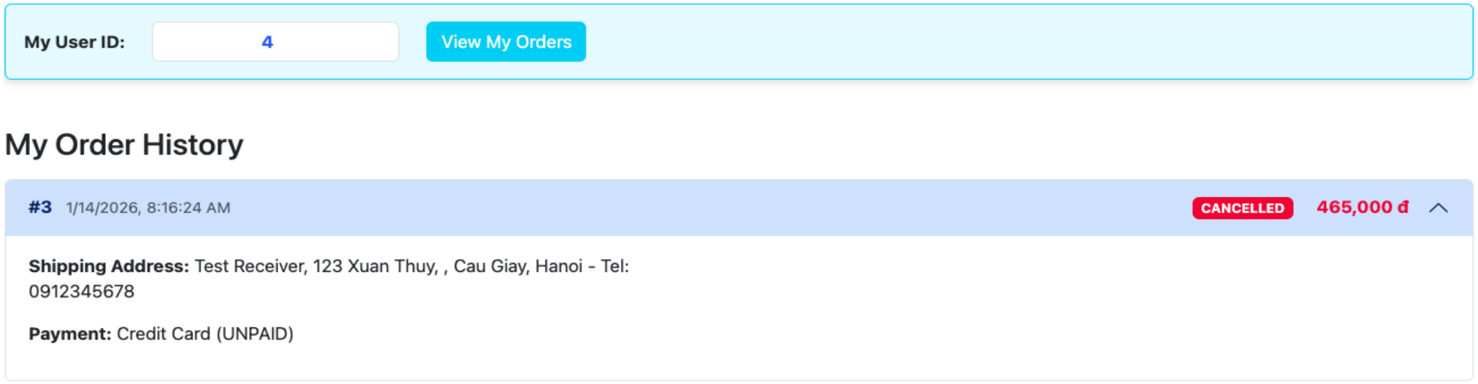


```
DECLARE @UserID INT; SELECT TOP 1 @UserID = id FROM users WHERE email LIKE 'flow_test_@gmail.com' ORDER BY id DESC;
DECLARE @OrderID INT; SELECT TOP 1 @OrderID = id FROM orders WHERE user_id = @UserID ORDER BY id DESC;

-- Create a snapshot of stock before cancel
DECLARE @StockBefore INT; SELECT @StockBefore = stock FROM product_variants WHERE id = 1;
PRINT CONCAT('Stock Before Cancel: ', @StockBefore);

-- Perform Cancel
EXEC cancel_order @UserID, @OrderID;
```

78 % — No issues found

Results | Messages

	result
1	SUCCESS: Order cancelled.



My User ID: 4 — View My Orders

My Order History

#3 1/14/2026, 8:16:24 AM — CANCELLED — 465,000 đ

Shipping Address: Test Receiver, 123 Xuan Thuy, , Cau Giay, Hanoi - Tel: 0912345678

Payment: Credit Card (UNPAID)

HUST

hust.edu.vn fb.com/dhbkhn

Perfomance Optimization & Benchmarking

V. Perfomance Optimization & Benchmarking (Index)

1. Indexes supporting customer features

Table	Index Name	Columns (Key + Include)	Optimization Goal
Products	`idx_products_active_newest`	Key: `(is_active, created_at DESC)` Include: `name`, `price`, `thumbnail`, `rating`	**Homepage / Browse**: Allows instant retrieval of "Newest Active Products" without touching the main heap.
ProductCategories	`idx_product_categories_category_id`	Key: `(category_id)` Include: `product_id`	**Category Filter**: Accelerates "Show all Men's Shirts" queries.
Products	`idx_products_price`	Key: `(price)` Include: `name`, `thumbnail`	**Price Filter**: Optimizes "Price Range" searches and "Sort by Price".
Products	`idx_products_name`	Key: `(name)`	**Name Search**: Supports basic keyword search on product names.

V. Perfomance Optimization & Benchmarking (Index)

1. Indexes supporting customer features

ProductVariants	`idx_product_variants_product`	Key: `(product_id)` Include: `color`, `size`, `stock`	**Product Detail**: Fetches all available sizes/colors instantly when viewing a single product.
Reviews	`idx_reviews_product_created`	Key: `(product_id, created_at DESC)` Include: `rating`, `comment`	**Review Display**: Loads latest reviews for a product without sorting cost.
Orders	`idx_orders_user_created`	Key: `(user_id, created_at DESC)` Include: `status`, `total_amount`	**My Orders**: Instant access to user's order history sorted by newest.
CartItems	`idx_cart_items_cart_id`	Key: `(cart_id)` Include: `variant_id`, `quantity`	**Cart View**: Rapidly joins cart items with product info.

V. Perfomance Optimization & Benchmarking (Index)

2. Indexes supporting admin features

Table	Index Name	Columns (Key + Include)	Optimization Goal
Orders	`idx_orders_created_at`	Key: `(created_at)` Include: `status`, `final_amount`	**Revenue Reports**: Scans order dates for daily sales reports without full table scan.
Orders	`idx_orders_status`	Key: `(status)` Include: `user_id`, `total_amount`	**Order Mgmt**: Quickly filters "PENDING" or "SHIPPING" orders for fulfillment workflow.
SupportMessages	`idx_support_mes sages_user`	Key: `(user_id)` Include: `content`, `status`	**Support History**: Quick lookup of a specific user's ticket history.
ProductCategories	`idx_product_categories_cat`	Key: `(category_id)` Include: `product_id`	**Analytics (Cat)**: Optimized for "Revenue by Category" reporting queries.
OrderItems	`idx_order_items_variant_id`	Key: `(variant_id)` Include: `quantity`, `price`	**Best Sellers**: Aggregates sales data per variant efficiently.

HUST

hust.edu.vn fb.com/dhbkhn

Triggers

VI. Triggers

Trigger Name	Table	Event	Description
`trg_prevent_negative_stock`	`product_variants`	AFTER UPDATE	**Inventory Safety Net.** Prevents stock from going below zero. If any UPDATE results in `stock < 0`, the trigger rolls back the transaction and raises an error. Protects against manual SQL updates bypassing checkout logic.
`trg_audit_order_status_change`	`orders`	AFTER UPDATE	**Order Status Monitor.** Prints status transitions to the Messages tab when order status or payment status changes. Output format: AUDIT: Order #ID status changed: OLD -> NEW | Payment: OLD -> NEW. Useful for real-time debugging during development.
`trg_address_single_default`	`addresses`	AFTER INSERT/ UPDATE	**Single Default Address.** Ensures only ONE address per user can be default. When `is_default=1` is set, automatically unsets all other addresses for the same user.
`trg_user_payment_single_default`	`user_payments`	AFTER INSERT/ UPDATE	**Single Default Payment.** Same logic as above for payment methods. Essential since no stored procedure manages this table.

HUST
ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

HUST

hust.edu.vn fb.com/dhbkhn

Conclusion and Future Work

VII. Conclusion and Future Work

1. Conclusion

Key Technical Deliverables

- **Stability:** Guaranteed by strict FK constraints and 3NF normalization.
- **Integrity:** Secured via encapsulation of Checkout & Inventory logic.
- **Performance:** Sub-second responsiveness (~10ms) powered by Covering Indexes.
- **Automation**: Database triggers enforce critical safety rules automatically, reducing the risk of human error.
- **Result:** A robust, secure, and scalable E-commerce backend.

VII. Conclusion and Future Work

2. Future Work

Future Enhancements & Upgrades

- **High-Speed Delivery:** Using **Redis** to reduce latency and handle flash-sale surges.
- **Fintech Integration:** Real-time processing with **VNPay/PayPal** for a seamless user experience.
- **Scalable Infrastructure:** Moving toward a **Microservices** architecture to support independent service growth.

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !