Manual Test Cases for E-Commerce API

Prerequisites

- Base URL: `http://127.0.0.1:8000` (or your configured host)
- Testing tool: Postman, curl, or any HTTP client
- Required headers: `Content-Type: application/json`
- Authentication: JWT tokens in `Authorization: Bearer {token}` header

Test Setup

1. Ensure the Django server is running
2. Database should have some initial data (products, categories)
3. Create test user accounts for different scenarios

---

1. AUTHENTICATION ENDPOINTS

Test Case 1.1: User Registration (POST /api/auth/register/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/auth/register/`
3. **Headers:** `Content-Type: application/json`
4. **Request Body:**

```
{
    "email": "testuser@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "password": "securepassword123",
    "phone_number": "+1234567890"
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "email": "testuser@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "phone_number": "+1234567890"
}
```

- **Verification:** Password field should not be returned
- **Side Effect:** Cart should be automatically created for the user

Error Test Cases:

- **Missing email:** Expected 400 with error message
- **Duplicate email:** Expected 400 with validation error
- **Invalid email format:** Expected 400 with validation error

Test Case 1.2: User Login (POST /api/auth/login/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/auth/login/`
3. **Headers:** `Content-Type: application/json`
4. **Request Body:**

```
{
    "email": "testuser@example.com",
    "password": "securepassword123"
}
```

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
}
```

- **Verification:** Both tokens should be valid JWT strings

Error Test Cases:

- **Invalid credentials:** Expected 401 Unauthorized
- **Missing fields:** Expected 400 Bad Request

Test Case 1.3: Token Refresh (POST /api/auth/refresh/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/auth/refresh/`
3. **Headers:** `Content-Type: application/json`
4. **Request Body:**

```
{
    "refresh": "{refresh_token_from_login}"
}
```

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
}
```

Test Case 1.4: Get User Profile (GET /api/auth/profile/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/auth/profile/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "id": 1,
    "email": "testuser@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "phone_number": "+1234567890"
}
```

Error Test Cases:

- **No token:** Expected 401 Unauthorized
- **Invalid token:** Expected 401 Unauthorized

2. CATEGORY ENDPOINTS

Test Case 2.1: List Categories (GET /api/categories/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/categories/`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "count": 5,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "name": "Electronics",
            "slug": "electronics",
            "description": "Electronic devices and gadgets",
            "image": null,
            "is_active": true,
            "created_at": "2024-01-01T00:00:00Z",
            "updated_at": "2024-01-01T00:00:00Z"
        }
    ]
}
```

Test Case 2.2: Create Category - Admin Only (POST /api/categories/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/categories/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {admin_access_token}`
4. **Request Body:**

```
{
    "name": "Books",
    "description": "Books and literature"
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 6,
    "name": "Books",
    "slug": "books",
    "description": "Books and literature",
    "image": null,
    "is_active": true,
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Error Test Cases:

- **Non-admin user:** Expected 403 Forbidden
- **Duplicate name:** Expected 400 Bad Request

3. PRODUCT ENDPOINTS

Test Case 3.1: List Products (GET /api/products/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "count": 10,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "name": "iPhone 15",
            "slug": "iphone-15",
            "category": 1,
            "category_name": "Electronics",
            "description": "Latest iPhone model",
            "price": "999.99",
            "discount_price": "899.99",
            "stock": 50,
            "is_available": true,
            "is_featured": true,
            "created_at": "2024-01-01T00:00:00Z",
            "images": [],
            "average_rating": 4.5,
            "get_discount_percent": 10
        }
    ]
}
```

Test Case 3.2: Filter Products by Category

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/?category=electronics`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** All returned products should belong to electronics category

Test Case 3.3: Filter Products by Price Range

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/?min_price=100&max_price=500`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** All returned products should have price between 100 and 500

Test Case 3.4: Search Products

Test Steps:

1. **Request Method:** GET

2. **URL:** `{base_url}/api/products/?search=iPhone`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Results should contain products with "iPhone" in name or description

---

4. CART ENDPOINTS

Test Case 4.1: Get User Cart (GET /api/cart/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/cart/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "id": 1,
    "user": 1,
    "items": [
        {
            "id": 1,
            "product": 1,
            "product_name": "iPhone 15",
            "quantity": 2,
            "total_price": "1798.00",
            "created_at": "2024-01-01T00:00:00Z"
        }
    ],
    "total_items": 2,
    "total_price": "1798.00",
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Test Case 4.2: Add Item to Cart (POST /api/cart-items/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/cart-items/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "product": 1,
    "quantity": 2
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 2,
    "cart": 1,
    "product": 1,
    "quantity": 2,
    "total_price": "1798.00",
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Test Case 4.3: Update Cart Item Quantity (PUT /api/cart-items/{id}/)

Test Steps:

1. **Request Method:** PUT
2. **URL:** `{base_url}/api/cart-items/1/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "quantity": 3
}
```

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Quantity should be updated to 3
- **Verification:** Total price should be recalculated

Test Case 4.4: Remove Item from Cart (DELETE /api/cart-items/{id}/)

Test Steps:

1. **Request Method:** DELETE
2. **URL:** `{base_url}/api/cart-items/1/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 204 No Content
- **Verification:** Item should be removed from cart

Test Case 4.5: Clear Cart (POST /api/cart/clear/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/cart/clear/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "message": "Cart cleared successfully"
}
```

---

5. ADDRESS ENDPOINTS

Test Case 5.1: Create Address (POST /api/addresses/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/addresses/`
3. **Headers:**
   - `Content-Type: application/json`
   - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "address_type": "shipping",
    "street_address": "123 Main St",
    "apartment_address": "Apt 4B",
    "city": "New York",
    "state": "NY",
    "country": "USA",
    "postal_code": "10001",
    "is_default": true
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "user": 1,
    "address_type": "shipping",
    "street_address": "123 Main St",
    "apartment_address": "Apt 4B",
    "city": "New York",
    "state": "NY",
    "country": "USA",
    "postal_code": "10001",
    "is_default": true
}
```

---

6. ORDER ENDPOINTS

Test Case 6.1: Create Order (POST /api/orders/)

Prerequisites: User must have items in cart and valid addresses

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/orders/`
3. **Headers:**
   - `Content-Type: application/json`
   - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "shipping_address_id": 1,
    "billing_address_id": 1,
    "shipping_cost": "10.00",
    "notes": "Please deliver after 6 PM"
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "order_number": "ORD-2024-001",
    "user": 1,
    "shipping_address": 1,
    "billing_address": 1,
    "order_status": "pending",
    "payment_status": "pending",
    "shipping_cost": "10.00",
    "total_price": "1808.00",
    "items": [
        {
            "id": 1,
            "product": 1,
            "product_name": "iPhone 15",
            "product_price": "899.99",
            "quantity": 2,
            "total_price": "1799.98"
        }
    ],
    "payment_method": null,
    "payment_id": null,
    "notes": "Please deliver after 6 PM",
    "tracking_number": null,
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Test Case 6.2: List User Orders (GET /api/orders/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/orders/`
3. **Headers:**
   - `Content-Type: application/json`
   - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Should only return orders for the authenticated user

Test Case 6.3: Cancel Order (POST /api/orders/{id}/cancel/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/orders/1/cancel/`
3. **Headers:**
   - `Content-Type: application/json`
   - `Authorization: Bearer {access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "message": "Order cancelled successfully"
}
```

Error Test Cases:

- **Order already shipped:** Expected 400 Bad Request
- **Order not found:** Expected 404 Not Found

---

7. REVIEW ENDPOINTS

Test Case 7.1: Create Product Review (POST /api/products/{product_id}/reviews/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/products/1/reviews/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "rating": 5,
    "comment": "Excellent product! Highly recommended."
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "user": 1,
    "product": 1,
    "rating": 5,
    "comment": "Excellent product! Highly recommended.",
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Test Case 7.2: Get Product Reviews (GET /api/products/{product_id}/reviews/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/1/reviews/`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Should return all reviews for the specified product

---

8. PAYMENT ENDPOINTS

Test Case 8.1: Create Payment (POST /api/payments/create/)

Prerequisites: Valid order must exist

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/payments/create/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "order_id": 1,
    "payment_method": "credit_card"
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "user": 1,
    "order": 1,
    "payment_id": "PAY-ORD-2024-001",
    "amount": "1808.00",
    "currency": "USD",
    "payment_method": "credit_card",
    "status": "pending",
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

Test Case 8.2: Process Payment (POST /api/payments/process/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/payments/process/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "payment_id": "PAY-ORD-2024-001"
}
```

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Payment status should change to "completed"
- **Verification:** Order payment_status should change to "paid"
- **Verification:** Order order_status should change to "processing"

Test Case 8.3: Request Refund (POST /api/payments/refund/)

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/payments/refund/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {access_token}`
4. **Request Body:**

```
{
    "order": 1,
    "amount": "1808.00",
    "reason": "Product defective"
}
```

Expected Results:

- **Status Code:** 201 Created
- **Response Structure:**

```
{
    "id": 1,
    "order": 1,
    "payment": 1,
    "amount": "1808.00",
    "reason": "Product defective",
    "status": "pending",
    "refund_id": null,
    "created_at": "2024-01-01T00:00:00Z",
    "updated_at": "2024-01-01T00:00:00Z"
}
```

---

9. ADMIN DASHBOARD ENDPOINTS

Test Case 9.1: Get Dashboard Summary (GET /api/dashboard/summary/)

Prerequisites: Admin user authentication

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/dashboard/summary/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {admin_access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Response Structure:**

```
{
    "total_orders": 150,
    "total_revenue": "45000.00",
    "total_customers": 75,
    "total_products": 50,
    "order_status": {
        "pending": 10,
        "processing": 25,
        "shipped": 20,
        "delivered": 90
    }
}
```

Error Test Cases:

- **Non-admin user:** Expected 403 Forbidden

Test Case 9.2: Get Recent Orders (GET /api/dashboard/recent-orders/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/dashboard/recent-orders/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {admin_access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Should return the 10 most recent orders
- **Verification:** Orders should be sorted by creation date (newest first)

Test Case 9.3: Get Top Products (GET /api/dashboard/top-products/)

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/dashboard/top-products/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer {admin_access_token}`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Should return top 10 products by order count
- **Verification:** Products should be sorted by popularity

---

10. ERROR HANDLING TEST CASES

Test Case 10.1: Unauthorized Access

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/orders/`
3. **Headers:** `Content-Type: application/json` (No Authorization header)

Expected Results:

- **Status Code:** 401 Unauthorized
- **Response Structure:**

```
{
    "detail": "Authentication credentials were not provided."
}
```

Test Case 10.2: Invalid Token

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/orders/`
3. **Headers:**
    - `Content-Type: application/json`
    - `Authorization: Bearer invalid_token`

Expected Results:

- **Status Code:** 401 Unauthorized
- **Response Structure:**

```
{
    "detail": "Given token not valid for any token type",
    "code": "token_not_valid",
    "messages": [
        {
            "token_class": "AccessToken",
            "token_type": "access",
            "message": "Token is invalid or expired"
        }
    ]
}
```

Test Case 10.3: Resource Not Found

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/nonexistent-slug/`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 404 Not Found
- **Response Structure:**

```
{
    "detail": "Not found."
}
```

Test Case 10.4: Validation Errors

Test Steps:

1. **Request Method:** POST
2. **URL:** `{base_url}/api/auth/register/`
3. **Headers:** `Content-Type: application/json`
4. **Request Body:**

```
{
    "email": "invalid-email",
    "first_name": "",
    "password": "123"
}
```

Expected Results:

- **Status Code:** 400 Bad Request
- **Response Structure:**

```
{
    "email": ["Enter a valid email address."],
    "first_name": ["This field may not be blank."],
    "last_name": ["This field is required."],
    "password": ["This password is too short. It must contain at least 8 characters."]
}
```

---

11. PERFORMANCE TEST CASES

Test Case 11.1: Response Time

Test Steps:

1. Measure response times for all endpoints
2. Record response times for different payload sizes
3. Test with multiple concurrent requests

Expected Results:

- **API Response Time:** < 2000ms for most endpoints
- **Database Query Time:** < 500ms
- **Large List Responses:** < 5000ms

Test Case 11.2: Pagination

Test Steps:

1. **Request Method:** GET
2. **URL:** `{base_url}/api/products/?page=1&page_size=10`
3. **Headers:** `Content-Type: application/json`

Expected Results:

- **Status Code:** 200 OK
- **Verification:** Should return exactly 10 items (or less if fewer available)
- **Verification:** Response should include pagination metadata

---

Test Environment Checklist

- ☐ Server is running and accessible
- ☐ Database has test data
- ☐ Test user accounts created (regular and admin)
- ☐ All endpoints return expected status codes
- ☐ All endpoints return expected data structures
- ☐ Authentication works correctly
- ☐ Authorization works correctly (admin vs regular user)
- ☐ Error handling works correctly
- ☐ Data validation works correctly
- ☐ Pagination works correctly
- ☐ Filtering and searching work correctly

Tools and Resources

Recommended Testing Tools:

1. **Postman** - GUI-based API testing
2. **curl** - Command-line testing
3. **HTTPie** - Command-line HTTP client
4. **Newman** - Command-line Postman collection runner

Test Data Requirements:

- At least 2 user accounts (1 regular, 1 admin)
- At least 3 categories

- At least 10 products across different categories
- At least 5 addresses for different users
- Test payment data

Environment Variables for Testing:

```
 BASE_URL=http://127.0.0.1:8000
TEST_USER_EMAIL=testuser@example.com
TEST_USER_PASSWORD=testpass123
ADMIN_USER_EMAIL=admin@example.com
ADMIN_USER_PASSWORD=adminpass123
```