# E-Commerce Cart Management System

## 1. Introduction

This document outlines the design and implementation of a cart management system for an e-commerce platform. The system will handle user interactions with the cart, such as adding discounts or removing discounts, removing items, and updating quantities, while ensuring real-time synchronization between the frontend and backend.

## 2. Objectives

- Provide a seamless and responsive user experience for cart operations.
- Ensure data consistency between the frontend and backend.
- Implement secure, scalable, and efficient operations.

## 3. Key Features

- Update Quantity: Users can modify the quantity of items in their cart.
- Add/Remove discount code
- Remove Items: Users can remove items from their cart.
- Real-Time Updates: Changes are reflected immediately in the cart view.
- Error Handling: User-friendly error messages for invalid operations (e.g., exceeding stock availability).

## 4. Challenges

- Ensuring accurate stock validation to prevent overselling.
- Handling edge cases such as invalid product IDs, out-of-stock items, or expired sessions.
- Maintaining security against potential CSRF or tampering attacks.

## 5. System Design

### 5.1 Frontend

- HTML, CSS and JS.
- API Communication: Fetch for REST API calls to the backend.

### 5.2 Backend

- Framework: Django with Django REST Framework (DRF) for API endpoints.
- Language: Python
- Database: Sqlite to store cart and product data.
- Validation: Backend validation for stock and pricing.

### 5.3 API Endpoints

1. Update Quantity: `PATCH /api/cart/update/`

    - Request: `{ "product_id": 123, "quantity": 5 ….}`
    - Response: `{ "success": true, "cart": { ...updated cart data... } }`

## 6. Security Measures

- CSRF Protection: Ensure all API requests are CSRF-protected.
- Validation: Verify product IDs, user sessions, and quantities on the backend.
- Rate Limiting: Prevent abuse of cart API endpoints by limiting request rates.

## 7. Chosen Implementation Approach: Real-Time Updates

### Why Real-Time Updates?

The real-time update approach has been chosen for the following reasons:

1. Used by Amazon and by other ecommerces sites
2. Enhanced User Experience

    - Immediate feedback ensures the cart reflects the most up-to-date information (e.g., stock availability, applied discounts, and price changes).
    - Reduces user frustration by showing errors (e.g., insufficient stock) immediately rather than during checkout.
3. Accuracy

    - Keeps the frontend and backend synchronized at all times, ensuring customers see accurate product availability and pricing.
    - Eliminates the risk of discrepancies caused by delayed batch updates.

4. Inventory Management
    - Real-time stock tracking prevents overselling or displaying out-of-stock items as available, a critical feature for e-commerce sites.
5. Dynamic Pricing Support
    - Reflects dynamic pricing changes immediately, ensuring the user is aware of price adjustments.
6. Security and Fraud Prevention
    - Helps monitor suspicious activity in real time, such as large changes in cart value, enhancing fraud detection.

---

## Implementation Details

### Frontend to Backend Sync

- Use `fetch()`to send API requests for every cart action:
    - Update Quantity
    - Add or remove discount
- Handle real-time responses to update the frontend UI dynamically.

### Validation Process

1. Verify the CSRF token to secure POST requests.
2. Validate stock availability and pricing on the backend.
3. Return detailed error messages if issues occur (e.g., "Requested quantity exceeds stock").

### Error Handling

- Provide user-friendly error messages for network or server issues.
- Implement retry mechanisms for failed operations.

---

## Benefits of Real-Time Updates

1. Immediate Feedback
    - Users know if their requested quantity or action is valid right away.
2. Frontend Accuracy
    - Prevents out-of-sync data between the frontend and backend.

3. Scalability for Larger Projects
    - Provides practical experience in handling real-time, frequent requests, useful for scaling in the future.

## 8. Conclusion

The proposed e-commerce cart management system prioritises real-time synchronization, user experience, and data accuracy. This implementation provides robust security measures and focuses on scalability, but at the same time the system provides a seamless and reliable shopping experience for customers while addressing potential challenges effectively.