**Add to Cart Flow**

**Key Components Involved in Add to Cart Flow**

* **CartService / CommerceCartService:** Manages cart operations like adding, removing, or updating items.
* **StockLevelService:** Ensures stock availability.
* **PricingService:** Fetches and applies product prices.
* **CalculationService:** Recalculates cart totals (subtotal, taxes, discounts).
* **PromotionService:** Applies promotions and discounts (if any).
* **EventService / Kafka Integration:** Publishes events for further processing.

**Tasks and Flow of Add to Cart Operation**

1. **User Action:**
   * The user selects a product and clicks the "Add to Cart" button on the storefront.
2. **Front-end Call (OCC/Storefront Controller):**
   * If using a **traditional storefront** (like Accelerator), the request is sent to the **CartController**.
   * If using an **OCC API**, the corresponding POST /cart/entry endpoint is called.
3. **Session Handling:**
   * The cart is linked with the **session**. If a cart doesn’t exist, a new cart is created in the session.
   * For logged-in users, the cart might be persisted in the database to associate with their account.
4. **CartService / CommerceCartService:**
   * The request reaches the **CartService** or **CommerceCartService**.
   * **CommerceCartService** checks if:
     + The product exists.
     + The product has valid stock and pricing.
     + The requested quantity is allowed (e.g., based on minimum or maximum quantities).
5. **Product Availability Check:**
   * **StockLevelService** checks if the product is available in stock.
   * If not, it may throw an exception or return a failure message.
6. **Price Calculation:**
   * **PricingService** is called to fetch the product’s price, which may vary based on:
     + Catalog version (e.g., UK/US catalogs)
     + User groups or price promotions.
     + Currency and tax settings.
7. **Cart Entry Creation:**
   * **CartEntryModel** is created for the product and added to the cart.
   * If the same product is already present in the cart, the quantity is updated (incremented or replaced).
   * Each cart entry contains:
     + Product reference.
     + Quantity.
     + Unit price and total price.
     + Applied promotions (if any).
8. **Promotion Engine Execution (Optional):**
   * If promotions apply, **PromotionService** recalculates the cart totals.
   * Example: Discounts, free shipping, or bundle offers.
9. **Cart Calculation:**
   * **CalculationService** recalculates the cart totals, including:
     + Subtotal.
     + Taxes.
     + Shipping fees.
     + Discounts.
10. **Persistence (for Logged-in Users):**

* If the user is logged in, the cart is saved to the **database** (e.g., MySQL) using the **CartModel** and related models.
* If using guest checkout, the cart is saved temporarily in the session.

1. **Event Publication (Optional):**

* A **CartModificationEvent** may be triggered to inform other services about the cart change.
* If using **Kafka** or any event-driven architecture, cart events can be published asynchronously.

1. **Response to Frontend:**

* A response is sent back to the storefront or API client with:
  + Success or failure message.
  + Updated cart totals and item count.
  + Any promotional messages (e.g., “You qualify for free shipping!”).

1. **UI Update:**

* The frontend updates the **mini-cart widget** to reflect the new item count and total price.
* If using AJAX, the cart UI is updated without reloading the entire page.

**Checkout Flow**

**Typical Checkout Flow Steps:**

1. **Cart Review:** Customer reviews the cart and initiates checkout.
2. **Address Selection:** Customer selects or adds a delivery address.
3. **Delivery Mode:** Customer chooses a delivery option (e.g., standard shipping).
4. **Payment Selection:** Customer provides payment details or selects a saved method.
5. **Order Review & Place Order:** Customer reviews the order summary and confirms.
6. **Order Creation:** **OrderModel** is generated, and the order process is triggered.
7. **Payment Authorization:** PaymentService interacts with the gateway for payment processing.
8. **Order Confirmation:** An email is sent, and the **OrderProcess** begins.

**Key Components in the Checkout Flow**

**1. CartService / CommerceCartService**

* **Purpose:** Manages the current cart, ensuring all items, quantities, and prices are valid.
* **Tasks:**
  + Retrieves the cart data.
  + Validates product availability, stock, and prices.
  + Manages applied promotions, vouchers, or discounts.

**2. CheckoutService / CommerceCheckoutService**

* **Purpose:** Manages the checkout process, such as setting delivery addresses, shipping methods, and payment methods.
* **Tasks:**
  + Validates the selected delivery address and payment method.
  + Ensures the chosen delivery mode is available for the user's location.
  + Converts the **CartModel** to **OrderModel** at the final step.

**3. PaymentService / PaymentProvider Integration**

* **Purpose:** Manages payment processing by integrating with external payment gateways (e.g., PayPal, Adyen, Stripe).
* **Tasks:**
  + Validates the payment information provided by the user.
  + Processes the payment and generates a transaction (e.g., **PaymentTransactionModel**).
  + Handles 3D secure authentication, if needed.

**4. DeliveryService / DeliveryModeService**

* **Purpose:** Handles shipping options and ensures delivery addresses are valid.
* **Tasks:**
  + Provides available **Delivery Modes** (e.g., Standard, Express).
  + Calculates shipping costs based on the user's address, cart contents, and delivery mode.
  + Validates the delivery address format and availability.

**5. CalculationService**

* **Purpose:** Recalculates cart totals during checkout (subtotal, taxes, shipping fees, discounts).
* **Tasks:**
  + Ensures that prices and totals reflect current taxes, promotions, and delivery fees.
  + Runs recalculation when the delivery method or payment mode changes.

**6. PromotionService / VoucherService**

* **Purpose:** Manages applied promotions and voucher redemptions.
* **Tasks:**
  + Ensures eligible discounts are applied to the cart.
  + Validates user-entered vouchers and applies them to the cart total.

**7. StockLevelService**

* **Purpose:** Ensures product availability during checkout.
* **Tasks:**
  + Checks stock levels to ensure ordered quantities are available.
  + If stock is insufficient, raises an exception or displays an error.

**8. OrderService / CommerceOrderService**

* **Purpose:** Converts the cart into a confirmed order.
* **Tasks:**
  + Creates an **OrderModel** from the current **CartModel**.
  + Stores the order in the database with all relevant details (items, prices, addresses, etc.).
  + Links the payment transaction to the order.

**9. CustomerAccountService**

* **Purpose:** Manages customer data and ensures the account is associated with the order.
* **Tasks:**
  + Retrieves customer data (address, payment preferences, etc.).
  + Saves addresses or payment details for future use if applicable.

**10. OrderProcess / BusinessProcess Engine**

* **Purpose:** Triggers the **order fulfillment workflow** using the Hybris Process Engine.
* **Tasks:**
  + Starts the **OrderProcess** after order confirmation.
  + Coordinates sub-processes such as payment capture, stock allocation, and shipping.
  + Notifies the user via email (e.g., order confirmation).

**11. EventService / Kafka Integration**

* **Purpose:** Publishes order events (like **OrderPlacedEvent**) to notify other systems.
* **Tasks:**
  + Sends messages for further processing, such as stock updates or delivery tracking.
  + Integrates with external systems using **Kafka** or other messaging queues.

**12. EmailService / NotificationService**

* **Purpose:** Sends order-related notifications to the customer.
* **Tasks:**
  + Sends confirmation emails after order placement.
  + Sends delivery updates and payment status emails.

**Order Flow**

**Typical Order Flow Steps**

1. **Order Placement:**
   * User completes the checkout, and the **CartModel** is converted to **OrderModel**.
2. **Payment Authorization:**
   * **PaymentService** validates and authorizes the payment.
3. **Order Confirmation:**
   * **OrderProcess** is triggered, and the order confirmation email is sent.
4. **Stock Allocation:**
   * **StockLevelService** allocates inventory for the order.
5. **Fulfillment Process:**
   * **FulfillmentProcess** coordinates packaging and dispatch.
6. **Delivery and Shipment:**
   * **DeliveryService** assigns shipping and sends tracking updates.
7. **Payment Capture (if required):**
   * **PaymentService** captures the payment after order dispatch.
8. **Order Completion:**
   * Once delivered, the order is marked as completed.
9. **Event Publication:**
   * Events like **OrderCompletedEvent** are published for further processing.
10. **Customer Notifications:**

* Emails and SMS updates are sent at various stages (e.g., confirmation, shipment).

**Key Components in the Order Flow**

**1. CartService / CommerceCartService**

* **Purpose:** Converts the cart into an order once the user completes the checkout.
* **Tasks:**
  + Retrieves the final **CartModel** with all products, prices, and details.
  + Validates that the cart is complete (e.g., address, delivery mode, payment).
  + Triggers the creation of an **OrderModel**.

**2. OrderService / CommerceOrderService**

* **Purpose:** Creates the **OrderModel** from the cart.
* **Tasks:**
  + Maps **CartModel** data to **OrderModel**.
  + Stores the **OrderModel** in the database.
  + Links the order with customer, payment, delivery, and stock information.

**3. PaymentService / PaymentTransactionService**

* **Purpose:** Manages payment processing and transactions.
* **Tasks:**
  + Ensures the payment was successfully authorized.
  + Creates **PaymentTransactionModel** linked to the order.
  + Processes payment capture (if payment is deferred until order fulfillment).

**4. StockLevelService / InventoryService**

* **Purpose:** Manages product stock levels and allocations.
* **Tasks:**
  + Allocates stock for the order once confirmed.
  + Updates stock levels across warehouses.
  + Releases stock if the order is canceled or fails validation.

**5. DeliveryService / DeliveryModeService**

* **Purpose:** Handles shipping details for the order.
* **Tasks:**
  + Assigns the appropriate delivery mode to the order.
  + Calculates delivery costs and adds them to the order total.
  + Manages shipment tracking and delivery updates.

**6. BusinessProcess Engine (OrderProcess / FulfillmentProcess)**

* **Purpose:** Orchestrates the order workflow from confirmation to delivery.
* **Tasks:**
  + Initiates **OrderProcess** after order creation.
  + Coordinates stock allocation, payment capture, packaging, and dispatch.
  + Manages sub-processes like payment retries, delivery notifications, etc.
* **Key Processes:**
  + **OrderProcess:** Manages order confirmation and handover to fulfillment.
  + **FulfillmentProcess:** Manages stock picking, packaging, and shipment.

**7. EventService / Kafka Integration**

* **Purpose:** Publishes order events to notify external systems and trigger asynchronous processes.
* **Tasks:**
  + Publishes events like OrderPlacedEvent or OrderCompletedEvent.
  + Triggers external systems (e.g., ERP, warehouse management systems).
  + Sends updates to downstream services through **Kafka** or other event-driven platforms.

**8. PromotionService / VoucherService**

* **Purpose:** Manages promotions and discounts on the order.
* **Tasks:**
  + Ensures the correct application of discounts or vouchers to the order.
  + Validates that promotions are still active at the time of order creation.
  + Adjusts order totals based on the applied promotions.

**9. CustomerAccountService**

* **Purpose:** Manages customer-related information linked to the order.
* **Tasks:**
  + Associates the order with the customer’s account.
  + Provides order history and tracking information to the user.
  + Handles guest orders and associates them if the user creates an account later.

**10. CalculationService**

* **Purpose:** Recalculates prices, taxes, and delivery charges for the order.
* **Tasks:**
  + Ensures the final order total reflects all applied promotions, taxes, and delivery fees.
  + Re-runs calculations if any changes occur during the order process (e.g., address change).

**11. EmailService / NotificationService**

* **Purpose:** Sends notifications to the customer and internal teams.
* **Tasks:**
  + Sends order confirmation, shipping updates, and payment receipts.
  + Notifies internal teams (e.g., fulfillment, finance) about the new order.
  + Sends alerts in case of payment failures or delays.

**12. Fraud Detection Service (Optional)**

* **Purpose:** Detects suspicious activities and prevents fraudulent transactions.
* **Tasks:**
  + Runs fraud checks during order placement (e.g., unusual orders or payment patterns).
  + Flags or blocks orders that trigger fraud rules.
  + Notifies administrators if manual intervention is required.

Top of Form