**Case Study-Array based Farmington**

**Introduction:**

**Farmington** is a next-generation agriculture e-commerce platform designed to bridge the gap between traditional farming practices and modern digital commerce. It provides a unified digital marketplace where farmers, wholesalers, retailers, and consumers can seamlessly interact, trade, and manage agricultural products and services.

**Objectives of Farmington Agriculture E-Commerce**

1. **Digitize Agricultural Trade**
   * Provide a user-friendly online platform for farmers to list and sell their products.
   * Bridge the gap between rural farmers and urban consumers.
2. **Enhance Order Management Efficiency**
   * Ensure fast, reliable handling of customer orders using efficient data structures like arrays.
3. **Improve Accessibility & Transparency**
   * Allow real-time access to product availability, pricing, and delivery status
4. **Efficient Order Processing (O(1))** ✅ Goal: Quickly manage customer orders in an organized manner.  
   ✅ How? Uses stack-based storage for orders to ensure recent orders are processed first.  
   🔹 Example:

**Array-Based Order Processing Analysis**

In Farmington, order processing is handled using arrays (Python lists), which offer **ordered, index-based, and memory-efficient storage**. Arrays allow the system to implement **FIFO (First-In, First-Out)** logic which is essential for **perishable agricultural goods**, ensuring **freshness and fairness**.

**Challenges in Array-Based Order Management**

**Inefficient Deletion from the Front (FIFO)**

* **Problem: In Python, deleting the first element from a list (pop(0)) causes all remaining elements to shift.**
* **Impact: This makes it O(n) time, which is inefficient for large datasets.**
* **Use Case: If 10,000+ orders are present, processing one can lag the system.**

**Solution: Use collections.deque, which allows O(1) time for front removal (popleft()).**

**2.Slow Search Time**

* **Problem: Arrays require linear time (O(n)) to search for a specific order by ID.**
* **Impact: Slows down order cancellation, tracking, or manual adjustments.**

**Order Management Issues and Solutions**

**1. ❌ Duplicate Orders**

**Issue: Customers accidentally place the same order multiple times.**

**Solution:**

* **Add order ID verification using a unique identifier per session.**
* **Implement a confirmation screen before final submission.**
* **Use timestamp tracking to flag orders placed too quickly after each other.**

**2. ❌ Out-of-Stock Items**

**Issue: Orders get placed for products that are already sold out.**

**Solution:**

* **Implement real-time inventory tracking linked with order system.**
* **Automatically disable out-of-stock items from the customer interface.**
* **Notify admins when stock hits reorder threshold.**

**3. ❌ Delayed Order Fulfillment**

**Issue: Orders are not processed or shipped on time.**

**Solution:**

* **Set up an automated order priority system (e.g., by delivery deadline).**
* **Use a status tracking pipeline: *Pending → In Progress → Shipped → Delivered*.**
* **Alert the admin if an order remains too long in a certain status.**

**4. ❌ Order Data Inaccuracy**

**Issue: Wrong quantities, prices, or customer info lead to failed deliveries.**

**Solution:**

* **Implement input validation at every form stage (e.g., quantity > 0).**
* **Show a summary page before final submission.**
* **Use auto-filled, verified customer profiles to avoid typing errors.**

**5. ❌ Order Cancellation Complexity**

**Issue: Customers can’t easily cancel or modify orders.**

**Solution:**

* **Allow limited-time cancellation (e.g., 30 mins after placing the order).**
* **Add a “Modify Order” option before shipment.**
* **Clearly show cancellation policy on the platform.**

**Time Complexity analysis**

| **🔧 Operation** | **💬 Description** | **⏱️ Time Complexity** |
| --- | --- | --- |
| **Add Order (append)** | **Add a new order at the end of the array** | **O(1) (amortized)** |
| **Search Order by ID** | **Linear search through the array for a specific order** | **O(n)** |

**Advantages of Using Arrays in Order Management**

**1. Simple & Easy to Use**

* **Arrays (or lists in Python) are easy to declare, access, and manipulate.**
* **Great for beginners or simple systems that don’t need complex operations.**

**2. Index-Based Access**

* **You can access any order instantly using its index: orders[0], orders[5], etc.**
* **Fast read access (O(1)) when you know the position.**

**3. Efficient for Sequential Data**

* **Orders are typically stored in the order they are received.**
* **Arrays naturally preserve this insertion order.**

**4. Low Overhead**

* **Arrays don’t need additional pointers like linked lists or trees.**
* **Memory-efficient, especially for small or medium-sized datasets.**

**5. Easy Iteration**

* **Perfect for looping through all orders for reports, summaries, or batch processing.**
* **Efficient for actions like:  
  → Filtering by status  
  → Summing total quantities  
  → Generating invoices**

**Code:**

**orders = []**

**def add\_order(order\_id, customer, item, quantity):**

**order = {**

**"id": order\_id,**

**"customer": customer,**

**"item": item,**

**"quantity": quantity**

**}**

**orders.append(order)**

**print(f"Order {order\_id} added successfully.")**

**def view\_orders():**

**print("\nAll Orders:")**

**for order in orders:**

**print(order)**

**def search\_order(order\_id):**

**for order in orders:**

**if order["id"] == order\_id:**

**return order**

**return None**

**def update\_order(order\_id, new\_quantity):**

**for order in orders:**

**if order["id"] == order\_id:**

**order["quantity"] = new\_quantity**

**print(f"Order {order\_id} updated successfully.")**

**return**

**print(f"Order {order\_id} not found.")**

**def delete\_order(order\_id):**

**for i in range(len(orders)):**

**if orders[i]["id"] == order\_id:**

**del orders[i]**

**print(f"Order {order\_id} deleted.")**

**return**

**print(f"Order {order\_id} not found.")**

**add\_order(101, "Akash", "Tomatoes", 10)**

**add\_order(102, "Preethi", "Carrots", 5)**

**add\_order(103, "Prithvi", "Onions", 8)**

**view\_orders()**

**print("\nSearching Order ID 102:")**

**print(search\_order(102))**

**update\_order(102, 12)**

**view\_orders()**

**delete\_order(101)**

**view\_orders()**