A. System Architecture

I noticed several orders in the system where “Admin” appears as the customer under **Admin → Orders**. From my understanding, the system is currently built using a shared login architecture.

While I understand the benefits of this approach (such as maintaining a single user database, centralized authentication, and a unified session store) I foresee a few potential drawbacks that I believe should be considered.

I would like to discuss these concerns further to ensure better system security, audit clarity, traceability, and long-term maintainability.

1. If Admin and Vendors are allowed to make purchases as Customers when the system goes live it will create confusion such as reports mixing real customer activity with internal test/support activity, customer data will be contaminated and audit trails will be unclear.

2. For the shared login architecture we will need some role-permission based restrictions like:

   ```
   ROLE_PERMISSIONS = {
      CUSTOMER: ["checkout"],
      ADMIN: [],
      SUBADMIN: [],
      VENDOR: [],
      SUBVENDOR: []

   if (!hasPermission(user, "checkout")) {
   denyCheckout();
   ```

3. *(Please note it is not an email based restriction but a role based restriction. When new vendors or subvendors or subadmins are added they will create new emails but these restrictions will still work as they are role based.)*

4. This approach will essentially allow the Admins to test certain accepts of the storefront like testing the coupons, viewing categories and subcategories, add products to cart to check the

cart behavior, etc. This will allow Vendors to view the products, its details, its pricing after they are added to the storefront.

5. This will create a clear separation of responsibilities wherein:
   Admins manage the platform
   Vendors manage products and fulfill orders
   Customers purchase products