Livestock Backend Update Notes

Overview

I’ve decided to rework the system design after reviewing the security and practicality of user accounts for this service.

This project is intended to simulate a simple online ordering system, not a full user platform, so requiring users to create accounts adds unnecessary security risk.

Removal of User Accounts

User accounts have been fully removed.

Instead of accounts, the system now only requires:

- Email address (for delivery updates)
- Full name (for shipping label)
- Delivery address

This reduces:

- Unnecessary storing of personal data (now it is GDPR compliant)
- Authentication logic
- Potential attack surface (password leaks, account takeovers)

Overall, this makes the system simpler and more secure for its intended use.

GDPR & Personal Data Handling

The delivery address is now treated as sensitive data.

While the system still needs to process and store it:

- The address will be hashed to comply with GDPR principles
- The email and full name remain readable for order lookup and support

The goal here is to **store only what is actually needed in readable form** and protect what doesn't need to be displayed frequently.

Product System Changes

Products are now:

- Pulled dynamically from the database
- Rendered live via JavaScript
- Automatically updated when the database changes

This removes the need for hardcoded products and allows:

- Live stock updates
- Easy future expansion

Price at Purchase

A new field in the SQL database has has been added - "price_at_purchase" this is because:

- Product prices can change over time
- Orders should always reflect what the user actually paid
- This also allows future additions like shipping fees, discounts, or tax

For now, shipping logic is not implemented, but the structure is there to support it later.

Order Flow (Current State)

Current flow when a user clicks **Buy**:

1. Product data is injected into the modal via JavaScript
2. Only the required data is sent to the PHP processing page
3. The system simulates a completed purchase (no real payment gateway yet)

At this stage:

- No database writes are happening yet
- The PHP file simply receives and validates the data
- This is intentional to ensure correct data flow first

Hashing Decision Update

After reviewing the system again, I've decided **not to use hashing for most fields**.

Because:

- Orders are direct receipts of user input
- There are no user accounts
- In case of issues staff may have to verify order details manually

All order-related fields must remain readable so:

- Orders can be searched
- Delivery issues can be resolved
- User mistakes (wrong name/address) can be checked

Password hashing logic has been removed as it's no longer relevant to this system.

Current Limitations

- No real payment processing (simulated only)
- Orders are not yet written to the database

These will be addressed in the next version.