Questions

1. What would be your first improvements if you had more implementation time?

If I had more implementation time, one of my first priorities would be to implement a proper authentication and authorization system with different permissions, improve the data validation on both the front-end and back-end to ensure users can’t submit incorrect or unexpected values, this would help avoid common issues and improve the overall reliability of the system.

Refining the user experience to make it more dynamic, responsive and intuitive adding smooth animations, smart form behaviors (like auto-fill), and faster, more responsive search experience.

With more time and information I could develop more customizable and context-aware data visualizations (graphs/charts). And also optimize performance, particularly database queries and large-data handling, to ensure scalability and speed as the system grows."

2. What approach would you use to insert a price history into the database and chart for the dashboard?

To implement a price history feature and display it as a chart on the dashboard, I would start by creating a dedicated table in the database to store historical price changes. This table would include fields such as ‘product_id’, ‘old_price’, ‘new_price’, and ‘changed_at’ (timestamp of the change). Every time a product's price is updated, a new record would be inserted into this table to preserve the change history.

On the front-end, I would create an API route like ‘/products/{id}/price-history’ to fetch the price history data. Then, using a charting library such as Recharts, I’d render a line chart showing how the price of a product has changed over time. This would provide users with clear and visual insights into product pricing trends directly from the dashboard.

3. What changes would need to be made to support updates in the product categories to have a discount percentage so that whenever the discount percentage is changed, the new price would be reflected in all products of the same category?

To support product category updates that include a discount percentage, I’d modify the database schema by adding a discount_percentage column to the categories table, allowing each category to have an associated discount.

Then, to ensure that all products in a category reflect the updated discount, I’d implement a mechanism to recalculate their prices whenever the discount changes. This could be handled either with a database trigger (for automatic updates) or through the application logic in the API layer. For example, in SQLite, a trigger could be used to automatically adjust the product prices whenever the discount percentage is updated.

Additionally, it would have to create the category update endpoint ‘PUT /categories/{id}’ would need logic to recalculate and persist the new prices for all related products. On the front-end, I’d ensure that product listings display both the original and discounted prices clearly. This setup ensures consistent and transparent discount application throughout the system.