Verification & Validation – Supermarket FreshChoice BV case

Authors: Roko Mladinić, Daniella Namuli, Simeon Markov, Zed Minabowan & Kristiano Mizher

Introduction:

In this document we'll be talking about the verification & validation of our FreshChoice supermarket project. Before we get into the document, we want to explain what our definition of verification & validation is.

- **Verification** = us testing whether what we built actually works the way we planned it in the analysis, planning and design.
- **Validation** = others testing our website and telling us whether they experience it the way we intended it to be built.

We'll start off by talking about what we think our website will do and what we found out during our personal tests (verification). After that we'll summarize what others thought after testing our website (validation).

Verification

Feature	Brief Description	Passed
Real-time Stock Tracking	Allows store managers to monitor inventory levels in real-time to optimize stock quantities and prevent shortages.	YES (All the GRUD operations are working properly handles)
Inventory Updating	Enables employees to quickly adjust inventory records following deliveries or sales to ensure data accuracy.	Partially met (The database support role base access but not implemented). Admin could create categories and products with those categories.
Multi-Location Scheduling	Permits managers to assign and manage employee shifts across various store locations for efficient workforce distribution.	YES (Multiple locations of the market could be created with assigned employees with their shifts).
Online Ordering & Pickup	Provides customers the ability to place grocery orders digitally for expedited in-store pickup.	Partially (No payment feature implementation, only frontend component is created and placeholder for the cart).

Commented [ZM1]: @Mladinić,Roko R. & @Minabowan,Zed P.X.O. did the feature & brief descrption

Commented [MS2]: @Markov,Simeon S. Verified the features

Validation of functional requirements

A. Component Design

-Single resposibility: New components must be atomic. For example, a `ChartComponent` should only handle visualization, not data fetching.

- Configurability: New components, especially the `AdminDataTable`, must use parameters to customize appearance, data source, and actions, preventing code duplication across different admin views (e.g., stocks vs. shifts).

- **Composition:** The Admin Dashboard pages will be built by composing simple components (`SectionHeader`, `AdminDataTable`, `ChartComponent`) to achieve complex administrative layouts.

B. Data Flow

- **Two-Way Binding:** All form fields in the new administrative components must utilize two-way binding (`@bind`) for synchronous state management, aligning with existing practices like the `SearchBar` implementation.

- **EventCallbacks:** CRUD actions (Create, Update, Delete) initiated from the administrative components must use **`EventCallbacks`** to communicate state changes back to the hosting page or service, promoting clear parent-child communication.

Validation of non-functional requirements

Commented [ND3]: @Namuli,Daniella D.

1. Performance Requirement: The dashboard should handle extensive upcoming data without breaking down the application.

The performance requirement has been partially met. The design document shows strong front-end performance considerations, including optimized CSS, scoped styling, use of transform animations, responsive layouts, and efficient use of Grid and Flexbox. These choices improve load speed and ensure smooth rendering. However, the document does not address how the application handles large datasets, real-time updates, or backend load. There is no mention of database performance, API efficiency, caching, pagination, or how the dashboard will manage heavy incoming data. Therefore, while the user interface is performance-aware, system-level performance needs further validation.

2. Scalability

Requirement: The architecture has to be built upon without additional frustrations.

The scalability requirement has been partially met. The document demonstrates a scalable design structure through consistent component styling, reusable layout systems, clear spacing rules, and CSS variable usage. It also shows readiness for future front-end expansion by listing planned enhancements such as hero carousels, personalization features, and live stock updates. However, scalability from a technical or architectural perspective is not covered. There is no explanation of backend scalability, code organization for Razor pages, project structure, or how new features can be integrated without causing system issues. As a result, only the design-side of scalability is fulfilled.

Usability

Requirement: The UI/UX has to meet design conventions (layout principles, colors, etc.), making it easy for customers and staff to navigate throughout the app.

The usability requirement has been fully met. The document strongly emphasizes UI/UX principles, including color psychology, visual hierarchy, F-pattern layout design, card-based component structure, and clear call-to-action elements. The interface follows accessibility standards with WCAG AA compliance, proper contrast ratios, semantic HTML, ARIA labels, and keyboard navigation. The design is also fully responsive, adjusting smoothly across mobile, tablet, and desktop devices. These elements ensure an intuitive, visually consistent, and user-friendly experience for both customers and staff.

Conclusion

Commented [MS4]: @Markov,Simeon S Complete conclusion

After the verification of each feature mentioned in the MoSCoW 'must-have' and the validation on the requirements categorized in the analysis document. The following conclusion could be drawn: The application covers the core functionalities, keeping programming and design compliances, and clear reporting of the process.