Sprint-1 User story document

Sprint Document_RE-2-MongoDB Learning and Implementation

Document History

Author	Version	Date	Description	Comments
Giridharan Sankaran	**0.1**	**22-Jan-2024**	**Functionality and Technical document updated**	
Ashwin Krishna Mani	**0.2**	**02-Feb-2024**	**Test results document updated**	
Yogesharvind Nedumaran	**1.0**	**02-Feb-2024**	**Reviewed and Baselined**	

1. **Background**

In the dynamic realm of retail, the adept and systematic management of product information plays a pivotal role in ensuring operational efficiency, regulatory compliance, and the delivery of a seamless customer experience. In our current sprint, the primary objective is to construct a resilient database system utilizing MongoDB. This database will be instrumental in storing and managing vital details of retail products, with a special focus on automating the tracking and management of product expiry dates. The system will also encompass supplier information and secure login credentials for both managers and store clerks.

By leveraging MongoDB, we aim to create a sophisticated infrastructure that not only centralizes critical product data but also streamlines the process of monitoring and handling product expirations. This automation will mitigate the need for manual intervention, leading to increased efficiency, reduced errors, and enhanced overall operational effectiveness in retail shops.

2. **Existing Functionality**

Presently, within retail establishments, crucial tasks such as restocking inventory and checking product expiry dates rely heavily on manual efforts by in-store clerks. This manual process becomes increasingly time-consuming, especially given the vast inventory of over 3000 products. Shop managers currently use mobile touch computer devices to identify and reorder products based on stock levels. However, the process of identifying and removing expired products involves assigning clerks to specific aisles. These clerks manually inspect each product within their designated area, leading to a laborious and time-intensive operation. This manual approach not only poses challenges in terms of efficiency but also increases the likelihood of human errors in identifying and removing expired items.

Furthermore, in the context of numerous supermarket chains, accessing APIs and required details from their developer portals poses challenges. Since we lack access to these external resources, we propose the creation of our own database, serving as a prototype. Despite not having direct access to supermarket chain APIs or necessary developer portal details, our database will mimic the functionalities required for this project. This approach ensures that the implemented logics will seamlessly integrate if eventually applied to the software of respective supermarket chains.

3. **Proposed Functional Solution**

The proposed solution aims to address the challenges associated with manual processes in retail, specifically the labor-intensive tasks of scanning items for reordering and manually checking and removing expired products. To streamline these operations, we propose automating the reordering process when product quantities are low and implementing a system to track product expiry dates.

The automated ordering system will trigger when product quantities fall below a specified threshold, automatically placing orders with suppliers. Simultaneously, an alert system will monitor product expiry dates daily. If a product is nearing expiration, the system will notify the manager, who can then decide to apply discounts for quick sales or delegate the task of removing expired products to store clerks using a predefined list, eliminating the need for manual item-by-item searches.

To implement this solution, we are developing a MongoDB database prototype. Although access to existing store APIs is currently unavailable, the MongoDB database will serve as a comprehensive solution. It will store product information, including expiry dates, manage inventory, store supplier contacts details, and securely maintain login credentials for managers and store clerks. This database-driven approach will provide an efficient and scalable solution to automate and enhance retail operations.

4. Design Analysis and Solution:

4.1 Technical changes:

4.1.1 Design Analysis:

As part of the technical changes, the implementation will leverage MongoDB Atlas to establish a prototype database housing comprehensive details related to products within the retail shop. The proposed design entails the creation of a MongoDB database named "rapideye360," aligning with the application's nomenclature. Within this database, distinct collections such as "Inventory," "Suppliers," "clerklogin," and "managerlogin" will be established.

4.1.2 Inventory Collection:

The "Inventory" collection serves as a pivotal component, accommodating multiple documents with fields including Category, Name, Brand, Price, Quantity, UPCID, Date of Procurement, Date of Expiry, and a unique identifier labeled "Difference." Specifically designed to store product information, this collection includes crucial details such as the name, brand, and price of each product. Additionally, a primary key, ensuring uniqueness, is assigned to every product, accompanied by the dates of procurement and expiry.

4.1.3 Suppliers Collection:

The "Suppliers" collection is dedicated to preserving email and contact details of suppliers associated with each product, organized based on their respective brand names. This ensures a systematic record of supplier information, facilitating streamlined management.

4.1.4 Clerklogin and managerlogin Collections:

Two additional collections, namely "clerklogin" and "managerlogin," are established to manage login credentials for store clerks and managers, respectively. These collections feature fields for Email and Password, providing secure access control to authorized personnel.

4.1.5 Detailed Structure:

Within the "Suppliers" collection, fields such as Brand Name, Supplier Email, and Supplier Contact are incorporated, ensuring a comprehensive representation of supplier-related information. In "clerklogin" and "managerlogin" collections, the inclusion of Email and Password fields further enhances security and access control measures.

4.1.6 Data Creation and Access:

All documents within these collections are meticulously created using MongoDB Compass, facilitating easy access to data from the database. This structured approach not only enables efficient monitoring of product expiry but also aids in real-time tracking of product quantities, ensuring timely restocking and optimal inventory management.

This intricate database design, implemented through MongoDB Atlas, aligns with the application's requirements, providing a robust foundation for retail shop operations.

5. **Use Test Cases:**

S.NO	Test Scenario	Expected Results
1.	Check if the created database is present in Mongodb server	The database should be present
2.	Check if the created collection "Inventory" is present in Mongodb server	The database should be present
3.	Check if the created collection "Suppliers" is present in Mongodb server	The database should be present
4.	Check if the created collection "clerklogin" is present in Mongodb server	The database should be present
5.	Check if the created collection "managerlogin" is present in Mongodb server	The database should be present
6.	Insert a document into the inventory collection and check if all the fields are present properly	System should allow to create new document in the collection
7.	Display the document with field name and field value of Difference in Inventory	System should show only the records in the document
8.	Display the document with field name and field value of Price in Inventory	System should show only the records in the document
9.	Display the document with field name and field value of Brand name in Suppliers	System should show only the records in the document
10.	Delete the document that was created in Inventory and check if the record is getting deleted	System should allow to delete the document

11.	Sort the documents in Inventory with Date of procurement field between two dates and check the data	Data should be correct
12.	Check if able to edit all the fields in the Inventory collection	Data should be editable

6. **Test Results Document:** TestresultsDocument_Sprint1.docx

7. **Assumptions:**

8. **Exclusions (Specific Functionality that is not covered by this Development):**

9. **KHD:**

<To be attached>