T25 - Software Systems Architecture - M.EIC010

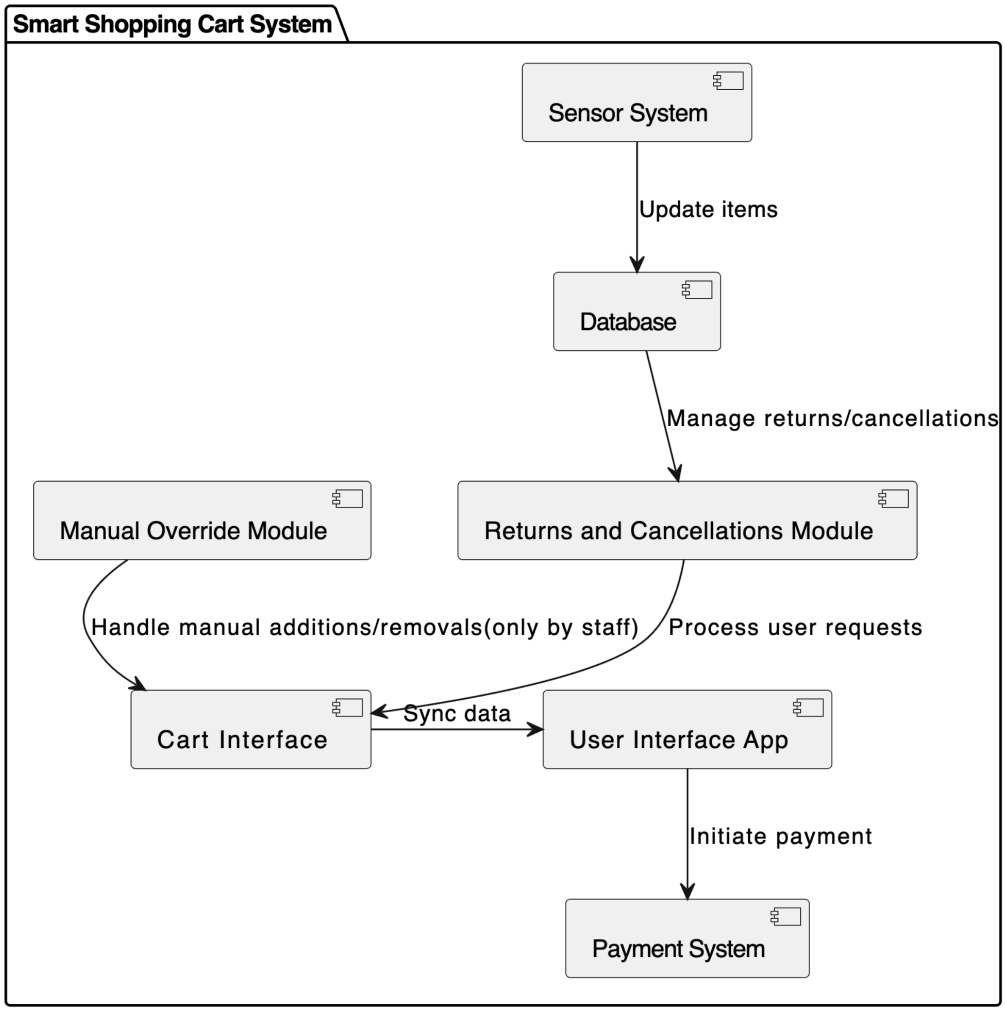
Rita Kiss - Laura Lumijärvi - Yana Peycheva - Juan José Osorio - Amanda Oliveira - Tiago Marques

Homework 08 - Automated Smart Shopping Cart

Introduction:

The Smart Shopping Cart system is a tool that's meant to speed up trips to the store. It keeps track of what clients put in their cart and adds up the cost as you shop. This documents shows some diagrams to explain how we would build this system, how shoppers wold use it, and what would they do if something goes wrong.

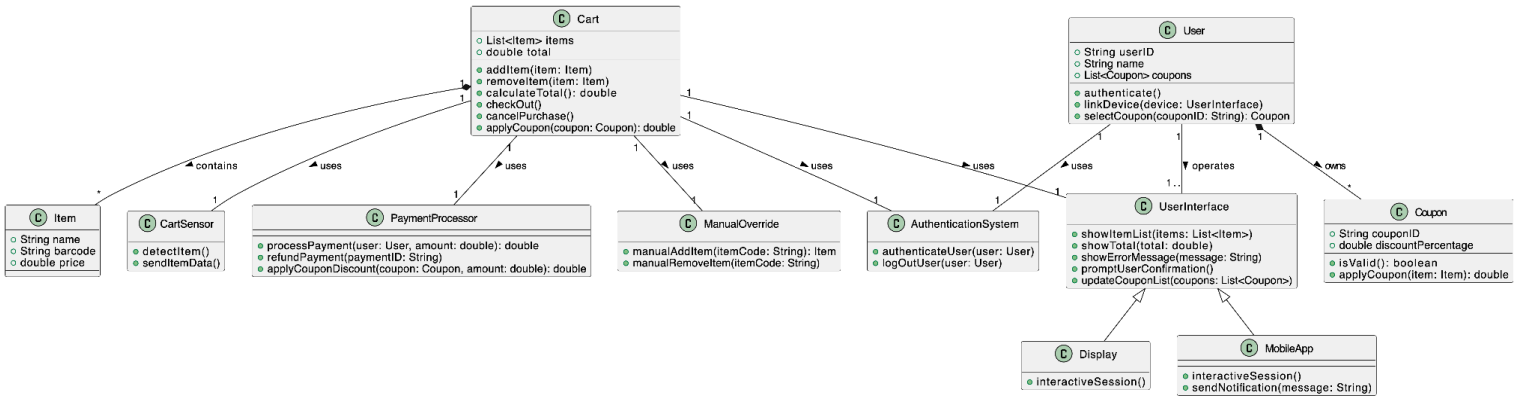
Components Diagram



Components Description:

Sensor System: Component responsible for detecting items that are added to or taken out of the cart. It updates the item list in real-time.
Database: Central storage that holds item data, user information, and transaction records. The Sensor System updates this database when items are added or removed from the cart. It also manages returns and cancellations, indicating the Returns and Cancellations Module.
Returns and Cancellations Module: Handles the processing of user requests regarding the return or cancellation of items of a previous purchase.
Manual Override Module: Allows staff to manually add or remove items from the cart, a control mechanism typically used for correcting errors or processing items that the Sensor System failed to detect.
Cart Interface: The interface of the cart (App or Display) that syncs data with the Manual Override Module and the User Interface App, acting as a mediator for data flow and user interaction.
User Interface App: Represents the application through which the user interacts with the cart, viewing item lists, running totals, and initiating payment.
Payment System: Processes payments initiated by the User Interface App, handling the transactional aspect of the checkout.

Class Diagram - Client Side



Classes and Functions Descriptions:

User: Includes a list of coupons. Handles authentication and can link to either the mobile app or the cart display.

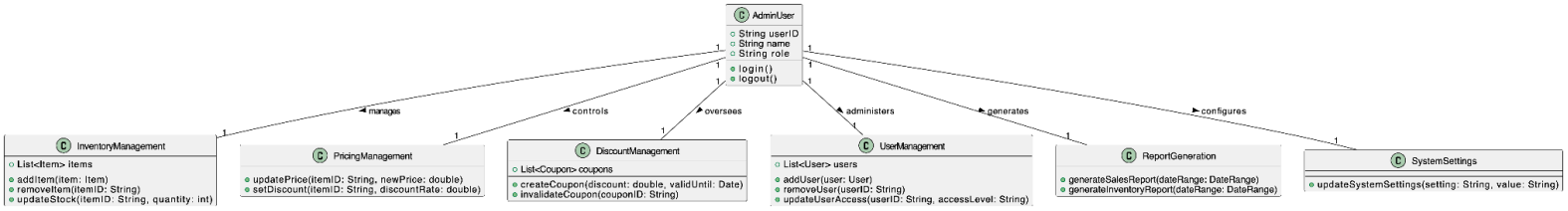
Coupon: Represents discount coupons with methods to check validity and apply discounts to items.

UserInterface: An abstract class extended by both **Display** and **MobileApp**, managing interactions such as item and total display, error messages, and user confirmations. Updates coupon lists for user review and application.

Display & MobileApp: Specialized forms of **UserInterface** for handling specific interactions depending on whether the user is using the attached cart display or their own mobile device. The mobile app can send notifications as well.

CartSensor: Measures and tracks the items that are added to or removed from the shopping cart.

ClassDiagram- Admin(Store) Side



Admin-Side Classes and Functions Descriptions:

AdminUser: Represents the administrative users (store managers, etc.) with functions to log in and log out.

InventoryManagement: Manages inventory details such as adding new items, removing items, and updating stock.

PricingManagement: Controls pricing for items, including the ability to update prices and set discounts.

DiscountManagement: Handles the creation and invalidation of coupons and discounts, managing their validity.

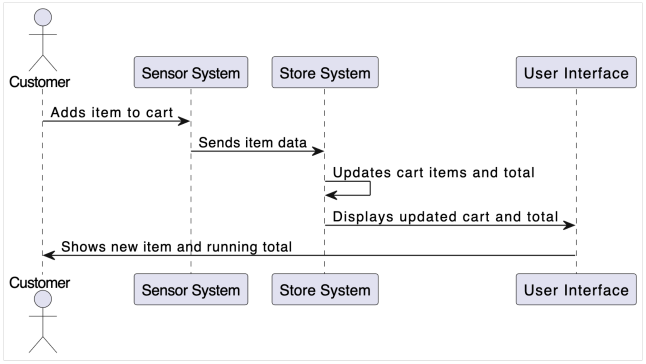
UserManagement: Manages user accounts, including adding and removing users.

ReportGeneration: Generates various reports for sales and inventory, supporting managerial decision-making by providing insights into business performance.

SystemSettings: Allows admins to update system settings, ensuring that configurations are maintained to match operational requirements or updates.

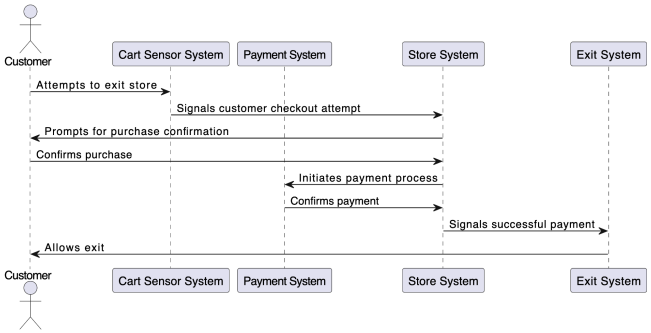
Scenarios(User Cases) and handling

Adding an Item to the Cart:



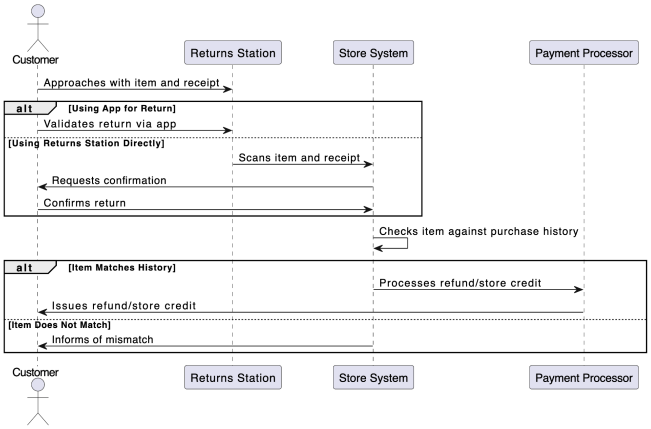
Scenario:
A customer selects an item and places it into the shopping cart.
Handling:
As soon as the item is placed into the cart, the cart's Sensor System detects the addition and communicates the item's data to the Store System. The Store System then updates the cart's item list and calculates the new total. This updated information is relayed to the User Interface, which displays the current list of items and the running total to the customer, either on the shopping cart's built-in display or the customer's mobile app.

Walking out of the Store (Checkout):



Scenario:
The customer completes their shopping and walks toward the store's exit with the intention to leave.
Handling:
The Cart Sensor System recognizes the customer's action and signals the Store System that a checkout attempt is occurring. The Store System prompts the customer, via the User Interface, to confirm their purchase. Upon confirmation by the customer, the Store System triggers the Payment System to process the payment using the customer's preferred method. After the payment is confirmed by the Payment System, the Store System communicates with the Exit System to allow the customer to leave. If the payment is not confirmed, the Store System alerts the customer to complete the necessary payment steps to finalize the purchase.

Returning a Previous Purchase:



Scenario: If a customer returns to the store to return an item from a previous purchase.

Handling: The user approaches a designated returns station in the store where they can scan the item along with a receipt or use their app to validate the return. The system checks the item against the purchase history and processes the return if it matches, refunding via the original payment method or offering store credit.

Quality Attributes and Architecture Handling:

1. Usability: The system provides an intuitive interface for customers to add and remove items, view running totals, and confirm purchases easily.
2. Reliability: The hardware components ensure accurate detection of items in the cart and reliable confirmation of purchases.
3. Security: Confirmation of purchase requires a PIN or other security measures to prevent unauthorized transactions.
4. Scalability: The system architecture can handle a large number of carts and customers simultaneously.
5. Maintainability: The modular design allows for easy maintenance and updates to the system components.
6. Performance: Efficient algorithms ensure fast processing of item additions, removals, and confirmation of purchases.
7. Flexibility: The system supports various payment methods and can adapt to changes in store inventory and pricing.