Faculdade de Engenharia da Universidade do Porto


# Software Systems Architecture

Homework #08 - Automated Smart Shopping Cart Architecture

**Team T23:**
Anete Pereira (up202008856)
Bárbara Carvalho (up202004695)
David Fang (up202004179)
Milena Gouveia (up202008862)
Pedro Correia (up202006199)

# 1. Introduction

In response to the evolving demands of modern retail, Automated Smart Shopping Cart aims at revolutionizing the traditional in-store shopping experience. By integrating sensor technology, real-time data processing and mobile app connectivity, the system focuses on streamlining the checkout processes, enhancing convenience and elevating security measures within retail environments. The development of this system started by establishing foundations, more specifically key quality attributes such as reliability, security, scalability, performance and interoperability. Through the architectural design, the priority is on the detection and tracking of items, secure payment processing, and dynamic inventory management.

# 2. System Overview

## 2.1 Data Types

- **Item:** Represents the individual products available for purchase in the store. Contains attributes such as item name, barcode, price, weight, aisle location, and availability status.
- **Cart:** Represents the physical shopping cart used by customers to store selected items. Contains information about the items currently in the cart, the total price, and the user associated with the cart.
- **User:** Represents individuals who interact with the system, either through the mobile app or by using the physical shopping cart. Contains user profile information, including name, contact details, preferences, discounts, and coupons.
- **Transaction:** Represents a completed purchase transaction initiated by a user. Contains details such as transaction ID, timestamp, total amount, payment method, and items purchased.
- **Payment Method:** Represents the different methods available for users to make payments. Includes credit/debit cards, digital wallets (e.g., Venmo), and any other accepted forms of payment.
- **Inventory:** Represents the store's inventory of items available for sale. Contains information about the quantity of each item in stock, as well as any associated attributes such as supplier details and reorder thresholds.
- **Notification:** Represents notifications sent to users to provide updates, alerts, or confirmations regarding their interactions with the system. Contains details such as notification type, content, recipient, and timestamp.
- **Security Event:** Represents security-related events triggered by the system, such as unauthorized access attempts or cart tampering. Contains details such as event type, location, timestamp, and any associated actions taken by the system.
- **Discount/Coupon:** Represents promotional discounts or coupons available to users for use during their shopping experience. Contains details such as discount amount, expiration date, applicable items, and redemption status.

## 2.2 Use Cases

### 2.2.1 User adds item to cart

When a user picks up an item and places it in the shopping cart, the cart's sensors detect the added item. Subsequently, the system updates the running total and the list of items in the cart. To ensure transparency and acknowledgment of the action, the cart screen device promptly displays a message or sends a notification to the app, confirming the addition of the correct item. This confirmation mechanism helps users verify that the intended item has been successfully added to their cart.

### 2.2.2 User removes item from cart

If a user decides to remove an item from the cart, the cart's sensors detect the removal of the specific item. Following this, the system updates the running total and the list of items in the cart accordingly. To maintain clarity and transparency throughout the shopping experience, the cart displays a message or sends a notification to the app, confirming the removal of the correct item. This confirmation ensures that users are informed about the successful removal of the intended item from their cart.

### 2.2.3 User views cart contents

When a user wishes to review the contents of their cart, they have the option to open the app or look at the cart's display screen. Upon accessing the cart contents, the app or display device provides a comprehensive view, including the running total of the items in the cart. Users can scroll through the list of items and access additional details such as quantity, price, and weight. This feature enables users to verify the accuracy of their selections and make any necessary adjustments before proceeding to checkout.

### 2.2.4 User confirms the purchase and leaves the store

Upon deciding to leave the store, the user initiates the purchase process either through their phone app or the cart's display device. To authorize the purchase, the user enters a PIN for authentication purposes. Additionally, the user selects their preferred payment method, which may have been previously set up during app initialization. The system securely processes the payment using the chosen method, such as a credit/debit card or Venmo. Once the transaction is completed, the user receives a digital receipt confirming the purchase. In the event that the user fails to confirm the purchase, an alarm is triggered to alert store staff, ensuring the security of the transaction.

### 2.2.5 User links cart to his phone

When a user with the app approaches a smart shopping cart, the cart's sensors detect the user's phone via Bluetooth or NFC technology. Following this detection, the cart and the user's phone establish a secure connection. This connection allows the system to link the user's profile, including any stored preferences, discounts, and coupons, to the cart. By linking the cart to the user's phone, the system can provide personalized shopping experiences and facilitate seamless transactions.

### 2.2.6 User leaves his cart

The system continuously monitors the cart's location, movements, and connection status with the user's phone. If a user leaves their own cart unattended, in the event of a disconnection due to an increase in distance, an alert is triggered, and a timeout period is initiated to restore the connection. If the connection is not restored within the specified timeout period, the system notifies store security to prevent potential theft or misuse of the cart.

### 2.2.7 User takes someone else’s cart

If a user attempts to connect to someone else's cart, the cart triggers an alert through the app or display device. Upon successful connection, the system recalculates the bill to reflect the items associated with the new user. This proactive alert system and recalibration process help maintain the integrity of transactions and prevent unauthorized access to carts.

### 2.2.8 User looks up for an item

To search for a specific item, the user utilizes the app or the cart's built-in interface. The user can enter the item's name or scan its barcode using the phone's camera or a dedicated scanner. Upon submission, the system retrieves relevant information about the item, such as its aisle location, price, and availability. Additionally, the user receives directions to the aisle where the item is located, streamlining the shopping process and enhancing user convenience.

### 2.2.9 The store sets prices, puts items on sale, etc

Store management has the capability to access the system and perform various administrative tasks, including setting prices for items and initiating sales. Discounts and electronic coupons are automatically applied based on user profiles or promotional campaigns. Furthermore, store management can adjust prices dynamically in response to factors such as demand, competition, or inventory levels. These administrative functions enable the store to optimize pricing strategies and enhance the overall shopping experience for customers.

### 2.2.10 The system synchronizes with the store’s inventory management system

The system maintains synchronization with the store's inventory management system to ensure real-time updates of inventory levels. As items are added or removed from the cart, the system deducts or adds them to the store's inventory database accordingly. Store management can monitor stock levels and make informed decisions regarding inventory replenishment based on sales data. This synchronization mechanism facilitates efficient inventory management and minimizes discrepancies between physical and recorded inventory levels.

## 2.3 Quality Attributes

### 2.3.1 Reliability

#### What does it mean in this domain?

- Reliability in this domain refers to the system's ability to accurately detect items being added or removed from the cart and maintain an updated running total. Failures in reliability could result from sensor malfunctions, software bugs, or connection errors.

#### How important is it?

- Reliability is crucial for customer trust and satisfaction. Failure to accurately track items or update totals could lead to incorrect billing, customer frustration, and loss of revenue for the store. While failures are possible, they should be rare and mitigated through rigorous testing of the system.

#### What does it do to the architecture?

- The architecture requires reliable hardware sensors, robust software algorithms, and fault-tolerant design patterns to ensure accurate detection and calculation. Redundancy in sensor systems and error handling mechanisms are essential components of the architecture.

### 2.3.2 Security

#### What does it mean in this domain?

- Security involves protecting user's personal and financial information, as well as ensuring the integrity of transactions. Failures in security could lead to unauthorized access, data breaches, or fraudulent transactions.

#### How important is it?

- Security is critical to protect sensitive information and maintain customer trust. A security breach could result in financial loss, legal repercussions, and damage to the store's reputation. Preventing security breaches is of utmost importance, but it can be challenging and expensive due to evolving threats and complex security requirements.

#### What does it do to the architecture?

- The architecture incorporates encryption protocols, secure authentication mechanisms, and access controls to safeguard sensitive data. It may require additional hardware components such as secure communication modules and tamper-resistant devices to enhance security measures.

### 2.3.3 Scalability

#### What does it mean in this domain?

- Scalability refers to the system's ability to handle increasing loads and accommodate a growing number of users and transactions. Failures in scalability could lead to performance degradation, long wait times, and system crashes during peak periods.

#### How important is it?

- Scalability is essential to ensure a smooth shopping experience, especially during busy hours or seasonal peaks. Failure to scale could result in lost sales opportunities, frustrated customers, and damage to the store's reputation. It may require significant investment and planning to achieve.

#### What does it do to the architecture?

- The architecture adopts scalable design patterns such as distributed processing, load balancing, and horizontal scaling to accommodate increased loads. It may require cloud-based infrastructure and elastic resources to dynamically adjust capacity based on demand fluctuations.

### 2.3.4 Performance

#### What does it mean in this domain?

- Performance relates to the system's responsiveness and efficiency in processing transactions, updating cart contents, and handling user interactions. Failures in performance could lead to slow response times, laggy interfaces, and degraded user experience.

#### How important is it?

- Performance is critical for customer satisfaction and retention. Slow response times or delays in transaction processing could frustrate users, leading to abandoned carts and lost sales. Optimizing performance requires efficient algorithms, resource allocation, and performance tuning efforts.

#### What does it do to the architecture?

- The architecture utilizes optimized algorithms, caching mechanisms, and efficient data processing techniques to minimize latency and maximize throughput. It may involve hardware upgrades, such as faster processors or solid-state drives, to improve system performance.

### 2.3.5 Interoperability

#### What does it mean in this domain?

- Interoperability involves the system's ability to integrate and communicate with external systems, such as payment gateways, inventory management systems, and other devices. Failures in interoperability could lead to data silos, communication breakdowns, and limited functionality.

#### How important is it?

- Interoperability is essential for easy operation and compatibility with existing infrastructure and services. Inability to integrate with external systems could result in manual workarounds, data inconsistencies, and missed opportunities for efficiency gains. Prioritizing interoperability enables the system to leverage existing investments and adapt to evolving business needs.

#### What does it do to the architecture?

- The architecture supports standardized interfaces, APIs, and data formats to facilitate integration with external systems. It may require middleware components, such as message brokers or service buses, to facilitate communication and data exchange between heterogeneous systems. Additionally, adherence to industry standards and protocols ensures compatibility and interoperability with third-party services and platforms.

# 3. System Architecture

For our architecture we had in mind the following architecture patterns:

- **Model View Controller**: The Application system likely follows the MVC architecture, where the user can interact with the View (UI), which communicates with the Controller (business logic), which in turn interacts with the Model (data). Additionally, the user can also interact dirrectly with the Controller, when adding/removing an item from the cart, which reports the changes to the View, right after communicating with the Model.
- **Event Driven**: The system employs an Event-Driven Architecture to facilitate loose coupling, asynchronous communication, and scalability. Events such as item additions or removals trigger updates to the system's data model and user interface components.
- **Layered**:The system can be seen as having multiple layers such as presentation layer (Application), business logic layer (example: Inventory Management System, Coupon Management System), and data storage layer (Models and Database). This promotes separation of concerns and maintainability.
- **Microservices Architecture**: The system is composed of multiple independent services each handling a specific functionality, promoting modularity and scalability.
- **Service-Oriented Architecture**: The system is composed of multiple services that communicate with each other to perform various functions. This architecture promotes loose coupling and reusability of services.
- **Client-Server Architecture**: The system involves a client-server architecture where the user interacts with the system through either the physical cart screen or the mobile application , while the system logic and data management are handled on the server-side.

## 3.1 Logical View

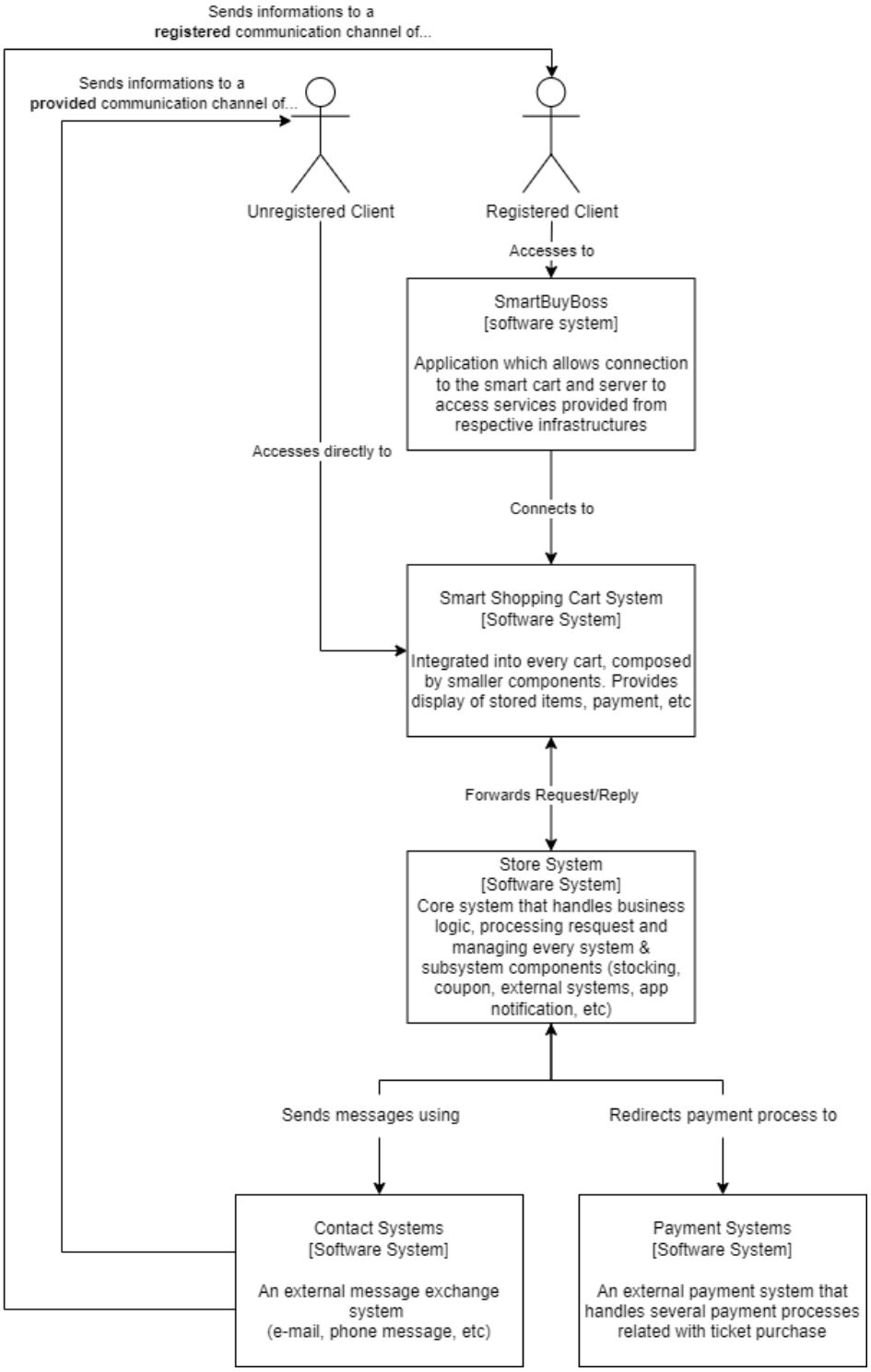



Figure 1: Context Diagram

The Context diagram provides a broad overview of the system architecture that allows for a comprehensive understanding of the system's scope. In this diagram, the systems are represented as a box, surrounded by its users and other interacting systems. The emphasis lies on identifying actors, roles, as well as software systems, rather than delving into technical

intricacies such as technologies and protocols. This zoomed-out view offers a high-level perspective suitable for communication with non-technical audiences.

In this scenario, we have two types of users: an unregistered client and a registered client. The registered client has access to the SmartBuyBoss application, which enables them to connect to the shopping cart. Through this application, they can see their discounts and view their purchase details on their smartphones. On the other hand, the unregistered user can only utilize the devices attached to the cart and cannot benefit from any discounts. The shopping cart system is then linked to the store system, allowing it to get information about the items. Furthermore, the store system is connected to both the contact system and the payment system. This allows users to confirm their payments using their preferred method and receive payment information on their preferred platform.

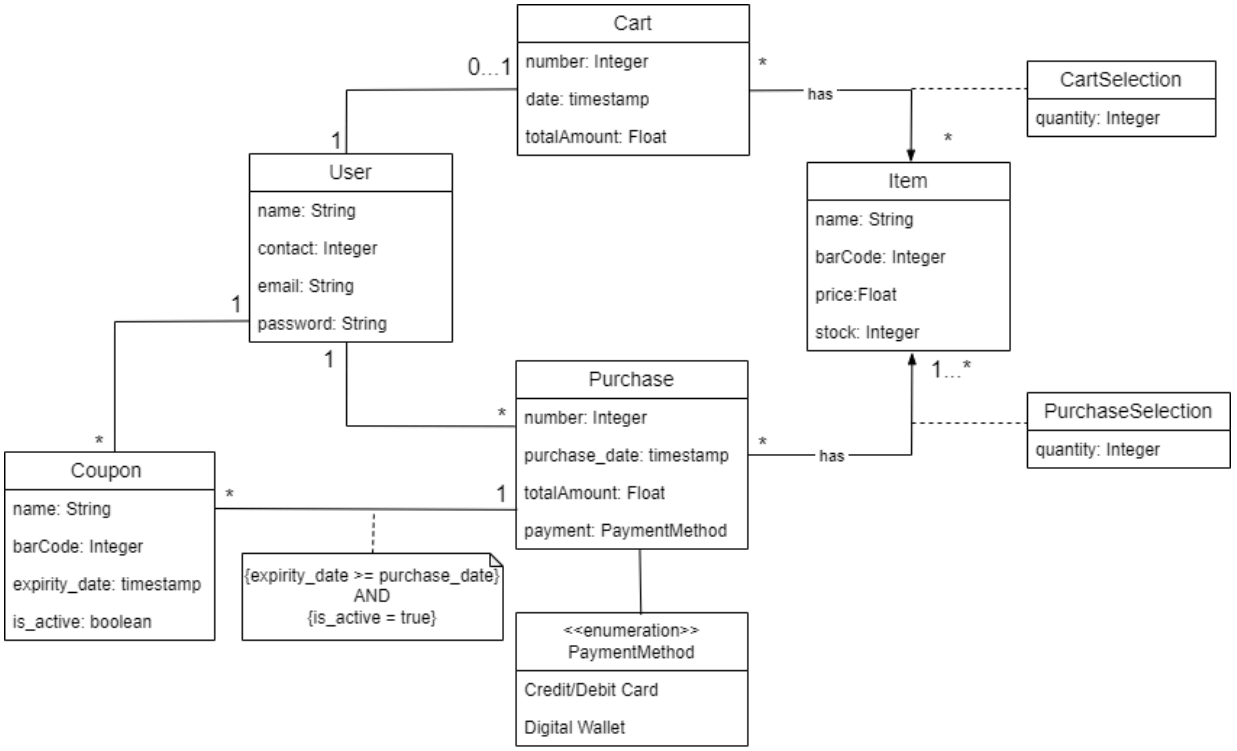
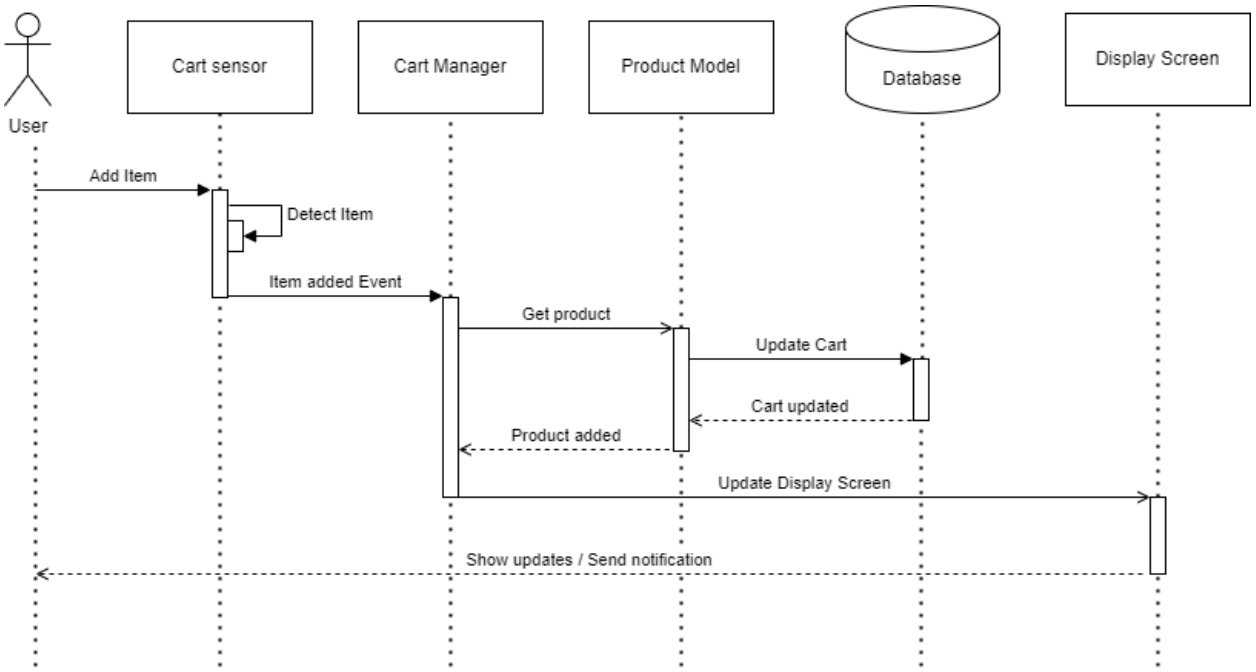



Figure 2: Logical View - Class diagram

The main classes of the system and the relationship among them are represented in Figure 2. The User can have at most one Cart and multiple Purchases. Both Cart and Purchase have a specific quantity of items selected. Additionally, the users have the option to use their coupons in purchases as long as they are active and within the expiration date.

## 3.2 Process View




| Name | Element | Description |
|---|---|---|
| Synchronous message symbol | ⟶ | This symbol is used when a sender must wait for a response to a message before it continues. |
| Asynchronous message symbol | ⟶ | Asynchronous messages don't require a response before the sender continues. |
| Asynchronous return message symbol | ⇠ - - - | Represented by a dashed line with a lined arrowhead. |
| Reply message symbol | ⇠ - - - | Represented by a dashed line with a lined arrowhead, these messages are replies to calls. |

Figure 3: Sequence diagram - Add item to cart

The figure above shows the actions the system does when a user adds an item to the cart. When the user adds an item, the cart sensor starts by detecting the item by finding the barcode, then sends the information of the item to be added to the cart. The information of the item is only shown after the cart is updated in the database. The “Display Screen” shown above can either be the screen of the phone or the device attached to the cart. If the user is using the application, the system also sends a notification to the user confirming the addition of the item.

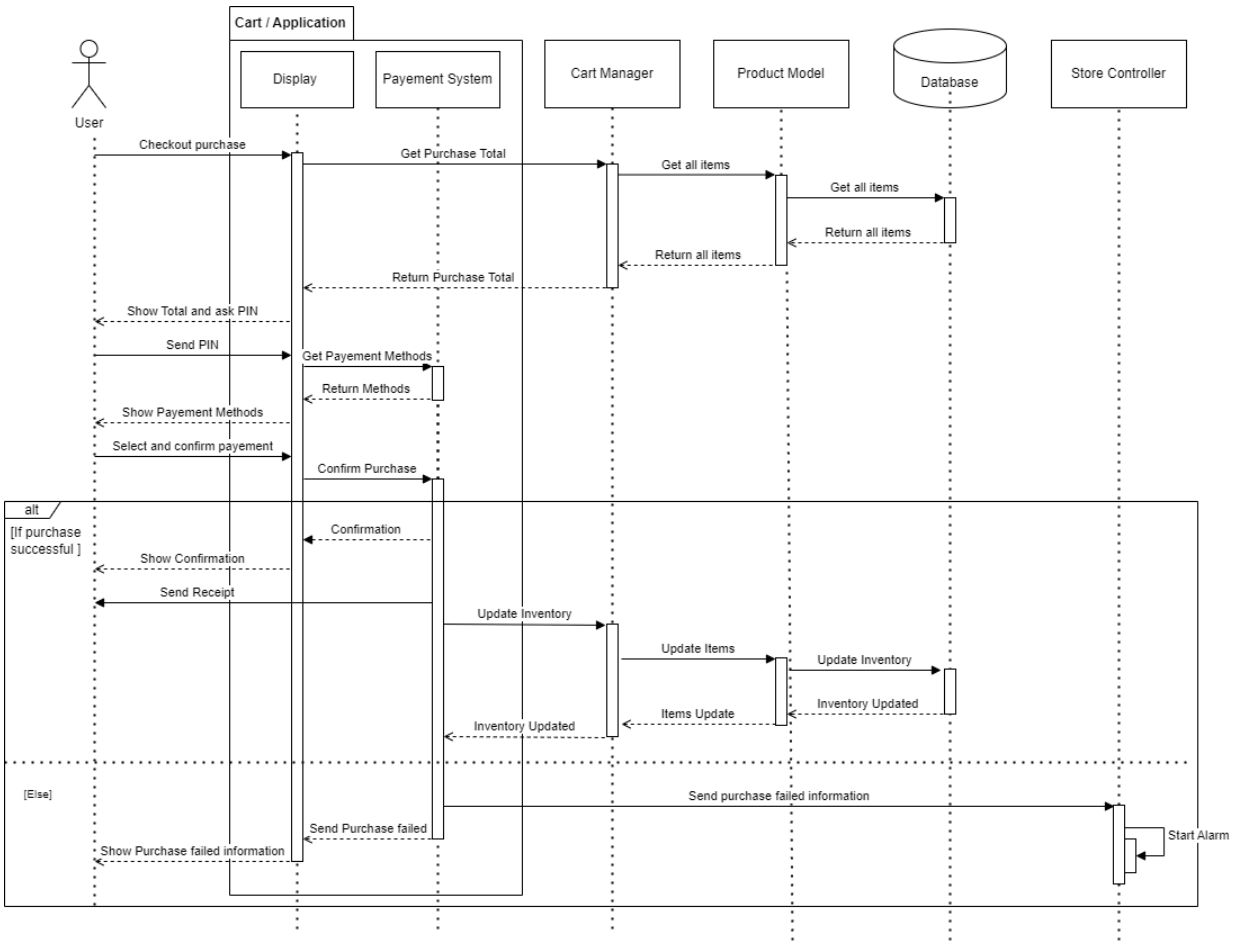
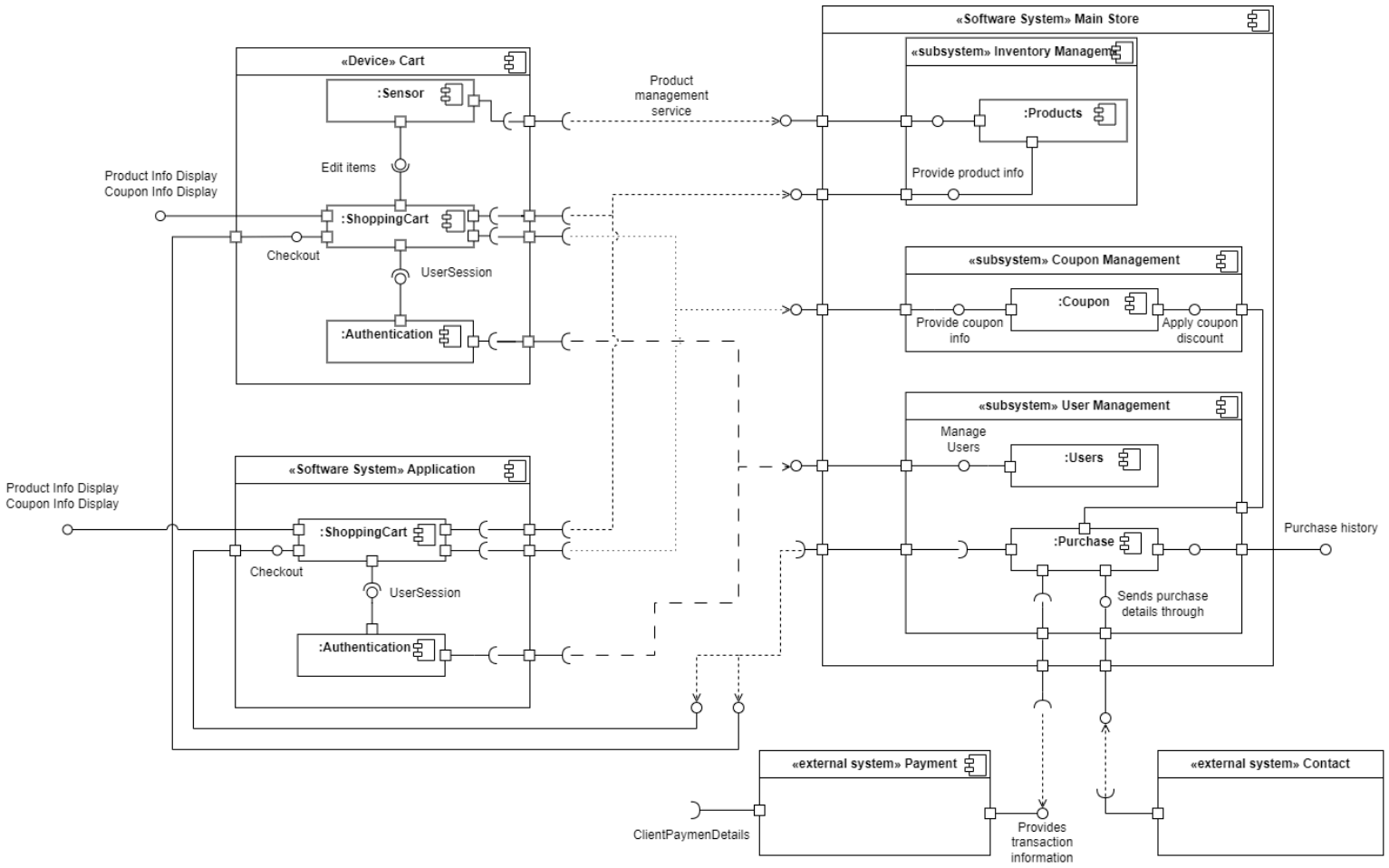



Figure 4: Sequence Diagram - Checkout

The figure above illustrates the checkout process initiated by a user. It begins with either the cart or the application retrieving the total amount to be paid by the user, including any discounts. Once the amount is displayed, the system prompts the user to enter their PIN. The user then selects the preferred payment method and completes the purchase. If the purchase is successful, a confirmation is displayed on the screen, the receipt of the purchase is sent to the user and the inventory is updated accordingly. If the purchase wasn't confirmed, an alert goes off to alert the store staff.

## 3.3 Implementation View




| Name | Element | Description |
|---|---|---|
| Port | □ | A port is often used to help expose required and provided interfaces of a component. |
| Require Interface | ——( | Required Interface symbols with only a half circle at their end (a.k.a. sockets) represent an interface that the component requires (in both cases, the interface's name is placed near the interface symbol itself). |
| Provided Interface | ——○ | Provided interface symbols with a complete circle at their end represent an interface that the component provides - this "lollipop" symbol is shorthand for a realization relationship of an interface classifier. |
| Dependency | -------→○ | A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. |

Figure 5: Component Diagram

A component diagram provides a high-level view of a system's architecture by illustrating the organization of system components and their dependencies. In our case, we'll focus on 7 major systems: 5 systems and subsystems within our application and two external systems.

### 3.3.1 Systems & Subsystems

- **Cart**: Represents the physical cart and its components, the sensor that makes it able to recognize what product and when a product is added to the cart, the shopping cart that stores every product added and their summed up price, and the authentication component that allows the user to authenticate into the system.
- **Application**: Refers to the users' virtual shopping cart, representing their virtual basket within the online application.
- **Inventory Management System**: Manages product functionalities such as changing the inventory, or giving product information.
- **Coupon Management System**: Manages the coupons and discount codes used by the client, when checking out, i.e, when purchasing the products on the shopping cart.
- **User Management System**: Provides user-related services for both login and registration of new users. Additionally, checkout operations are also available in this system, which are useful for confirming the payments, accessing the purchase history and managing user coupons.

### 3.3.2 External Systems

- **Payment System**: Represents the external payment gateways and integration which requires the user information to be able to complete the products purchase.
- **Contact System**: Manages communication with users (via e-mail, phone message, other), sending the purchase details, notifications/alerts about important states of the system.

## 3.4 Physical View

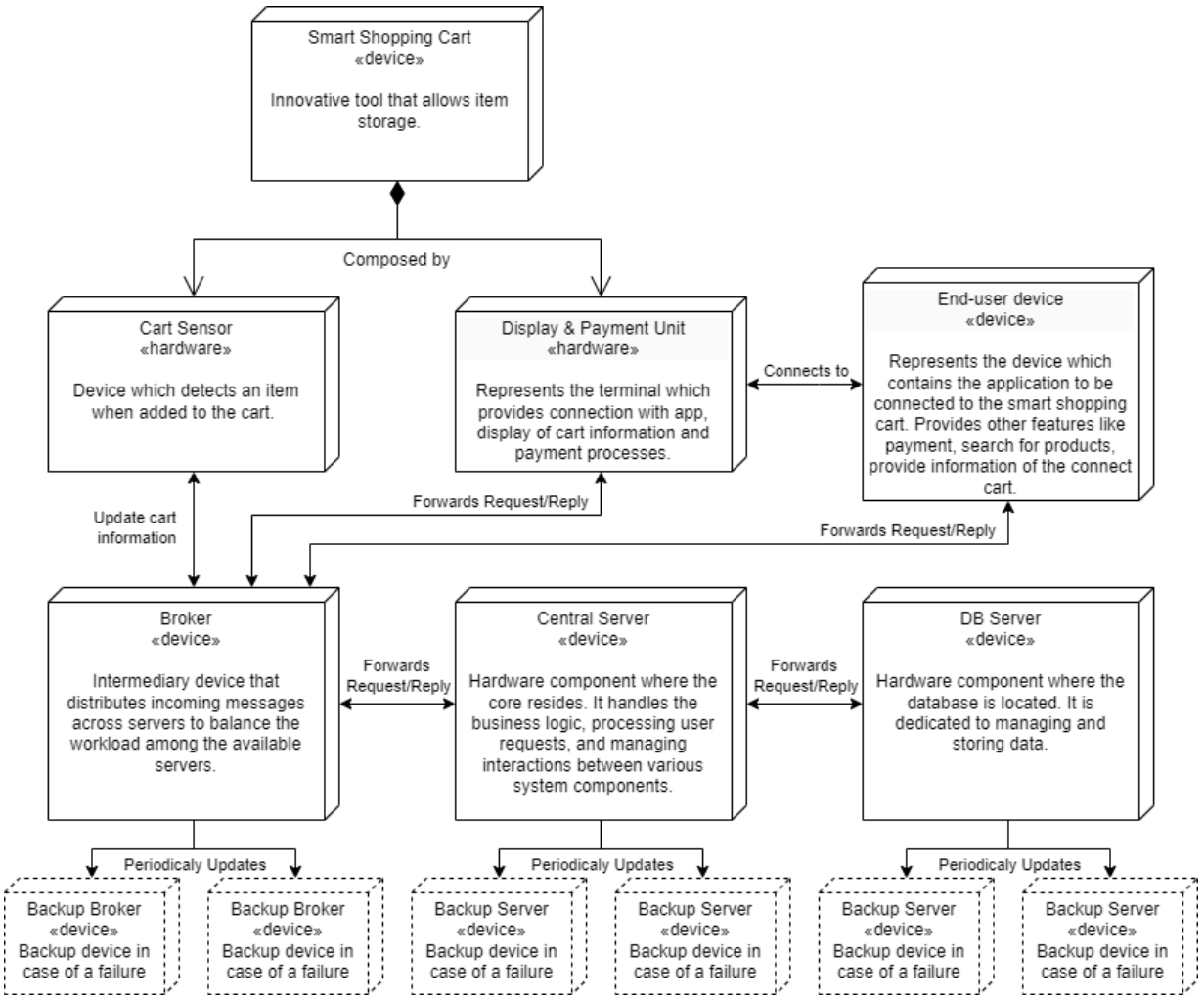
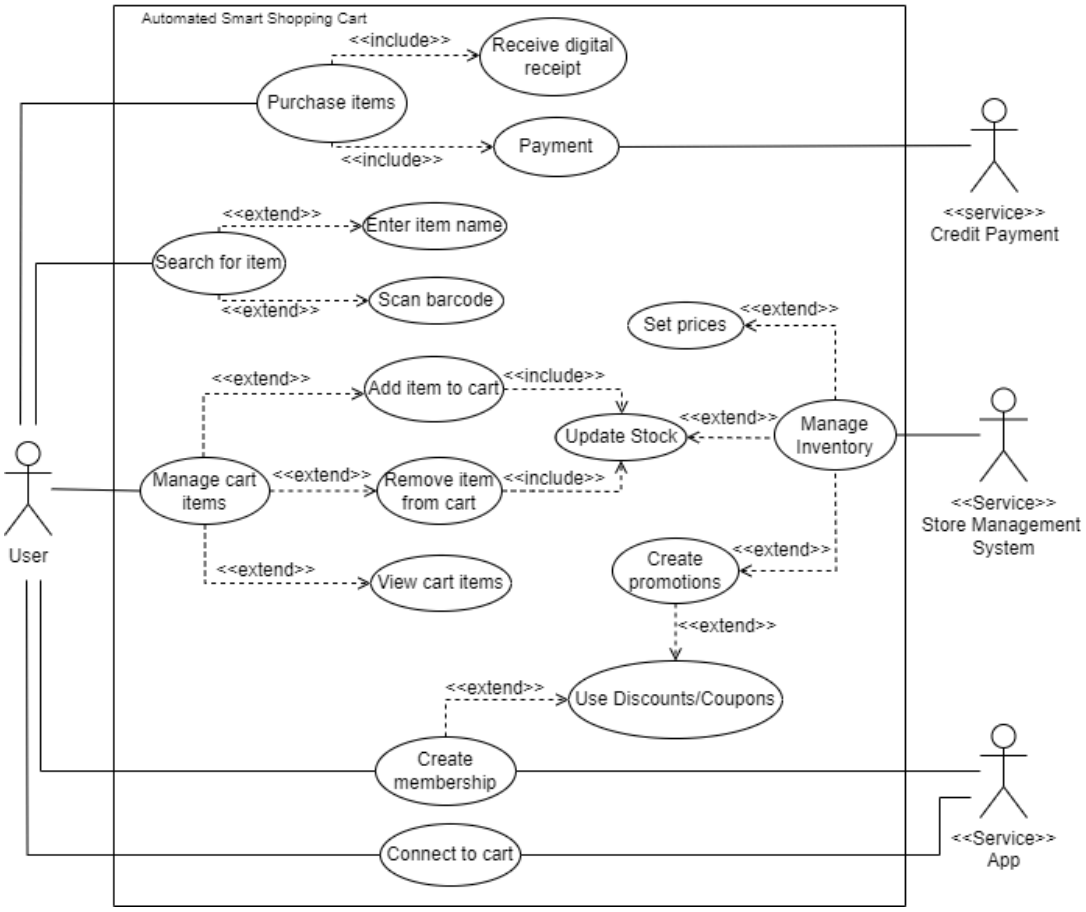



Figure 6: Deployment Diagram

The SmartBuyBoss system is deployed across several hardware components illustrated in the diagram above.

## 3.5 Use-Case View




| Name | Element | Description |
|---|---|---|
| Association | | A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases. |
| Include | | A line between use cases. One use case includes another use case means that the included use case is a part of the main use case and is essential for its execution. |
| Extend | | A line between use cases. One extended use case indicates one additional behavior of the main use case. It is optional. |
| Actor | | Each actor has a unique name that describes the role of the user who interacts with the system. |

Figure 7: Use Case diagram

In the use case diagram, the primary actor is the user who represents all shoppers interacting with the store, through the cart or app. There are also three other entities also important for the system: the payment service, the store management system and the app. An essential functionality of the system is the ability to link the smart shopping cart to the user's mobile phone and/or the mobile app. Furthermore, a user can join a membership to have access to discounts and coupons.

The main action a user can perform is to manage the cart items. This use case involves adding or removing items from the cart, which triggers the system to perform an update to the

store’s inventory. Additionally, the user can also view the contents of their shopping cart either through the mobile app or the cart’s display screen. One other important scenario is the ability to search for specific items within the store, using its name or the barcode.
To purchase the items on the cart, the user needs to select the payment method and authorize the payment. Upon confirmation, the system securely processes the transaction and emits a digital receipt.

Lastly, the store management system updates the inventory in real time, ensuring accurate stock management. It also is responsible for setting prices for items and creating promotional offers, such as discounts or coupons.

# 4. Conclusion

In conclusion, the architecture designed for the Automated Smart Shopping Cart project represents a carefully considered solution aimed at enhancing the shopping experience.

Emphasizing key qualities like reliability, security, and scalability, each decision was made with customer trust and satisfaction in mind. By incorporating robust hardware sensors and secure authentication mechanisms, the system ensures accurate item tracking and safeguards sensitive data. Furthermore, the adoption of scalable design patterns such as the Event-Driven Architecture and Microservices Architecture contributes to the system's adaptability and readiness for future changes, promising a seamless and dynamic retail experience.