

|  |
| --- |
| MIS 6308.0W1  **System Analysis**  **and**  **Project Management** Fall 2018 |

**GROUP 07:**

**AMOD KARAMBELKAR**

**APURVA SHERKE**

**KSHITIJ BHURE**

**NISHA DATE**

**SAMBIT SAHOO**

**CONTENTS**

**EXECUTIVE SUMMARY:**

Walmart Inc. is an American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores. The company was founded by Sam Walton in 1962 and is headquartered in Bentonville, Arkansas. Walmart is the world's largest company by revenue—over US$500 billion, according to Fortune Global 500 list in 2018. It is also the world’s largest private employer with 2.3 million employees. After studying the model and its various processes, our group has identified a couple of areas where the business processes of Walmart can be improved to increase revenue and customer satisfaction.  
  
To keep patrons engaged and remain one of the most frequented superstore chains in the country the corporation makes several efforts like rebranding and store design changes, environment initiatives, mobile applications, web ordering services. One other feature that Walmart offers is a 90-day return policy. Customers have the option to return goods purchased at Walmart for store credit or exchange within 90-days. A substantial number of customers are seen misusing this feature and frequently return goods after using them right up until the return deadline. The system improvement we are suggesting will be able to identify these customers that are seen returning items frequently and/or very close to the 90-day mark. This will help Walmart collect data on revenue lost due to the implementation of their return policy and take necessary action.  
  
Another improvement area we noticed is in their mobile application. Currently, when a customer searches for a product in a particular store the app is able to display the aisle number and shelf number where it can be found. However, it is difficult to navigate within the store and find these aisle numbers. Our new proposed system improvement will provide the consumer with in-store navigation. With the help of GPS services, the app will be able to direct the customer on how to reach the desired product location. This will save consumer’s time and improve customer satisfaction.

**Problem Statement:**

In the existing system, Walmart has a 90-day return policy and the returns processed are only tied to the Product ID which enables Walmart to identify whether an item was sold from their inventory. There is no system in place to tie it back to the customer. This results in people misusing their services and returning items often after using right till the 90-day mark. Return policies are for customer's satisfaction but any abuse in these policies should be tracked and an action should be taken against it.

Also, Walmart currently offers free delivery on orders that exceed $35. Each order can be comprised of several items to meet the free delivery eligibility threshold. A user can cancel an item once the order is placed and still avail free delivery. A customer should ideally be charged a shipping cost if the order value goes below $35.

Generally, Walmart stores are spread across a large area and it is difficult to locate a specific item inside the store. The existing feature of the Walmart app gives an information about the aisle number and shelf number of where a product is located. But, a feature that can provide dynamic navigation to the exact product location on shelf would prove to be very handy for the customers.

**Business Need:**

Below are the business needs which highlight the need for a new system

* Optimization of the current checkout delivery system based on returns by users
* Manage & keep records of customers returning product(s) very often
* Improvised utilization of Walmart app to users when they visit store

**Objectives of the proposed system are as follows**

* To mitigate or reduce the misuse of the 90-day return policy
* A system for recording the customer information tied to the returned product
* To project the statistics of returns made by a customer for Walmart
* Tracking system for verifying the order eligibility for free delivery before delivery of the product
* An in-house system for navigation within the store for the desired product in the App

**Scope**

The Scope of the new proposed system is as follow:

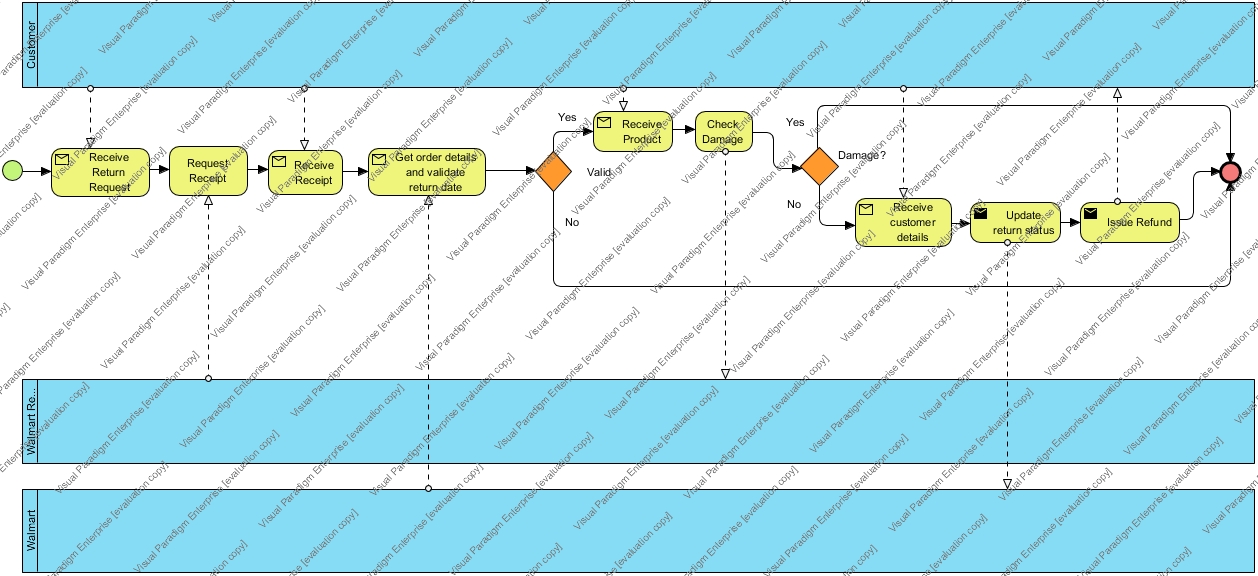
* The project would primarily focus on identifying the customers who are misusing the 90 days return policy and further take necessary actions
* New proposal would improve tracking system for the order eligibility and customers would be charged the shipping cost accordingly
* To enable In-store navigation for customers which would help them find the desired product location and improve customer satisfaction

**Context Diagram:**

CHOREOGRAPHY DIAGRAM:

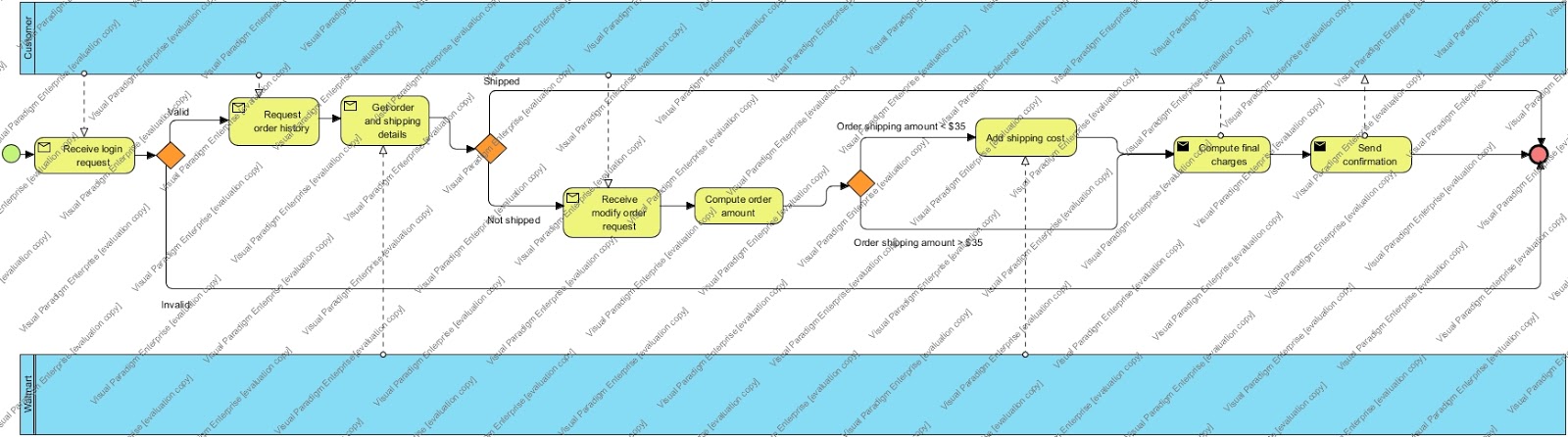
BPMN1:

Order Return at Store – Choreography Diagram

****

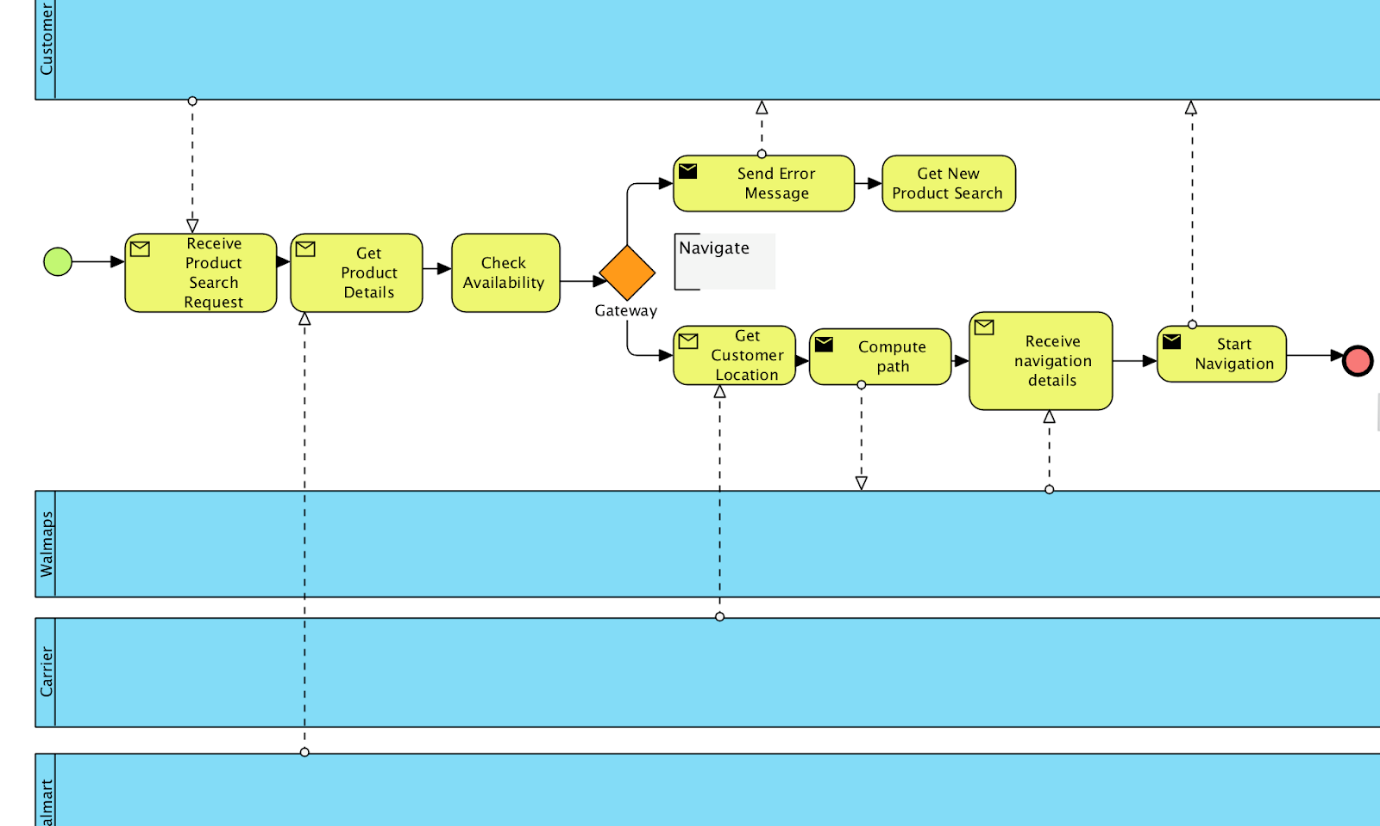
BPMN 2:

Cancel/Change Order – Choreography Diagram

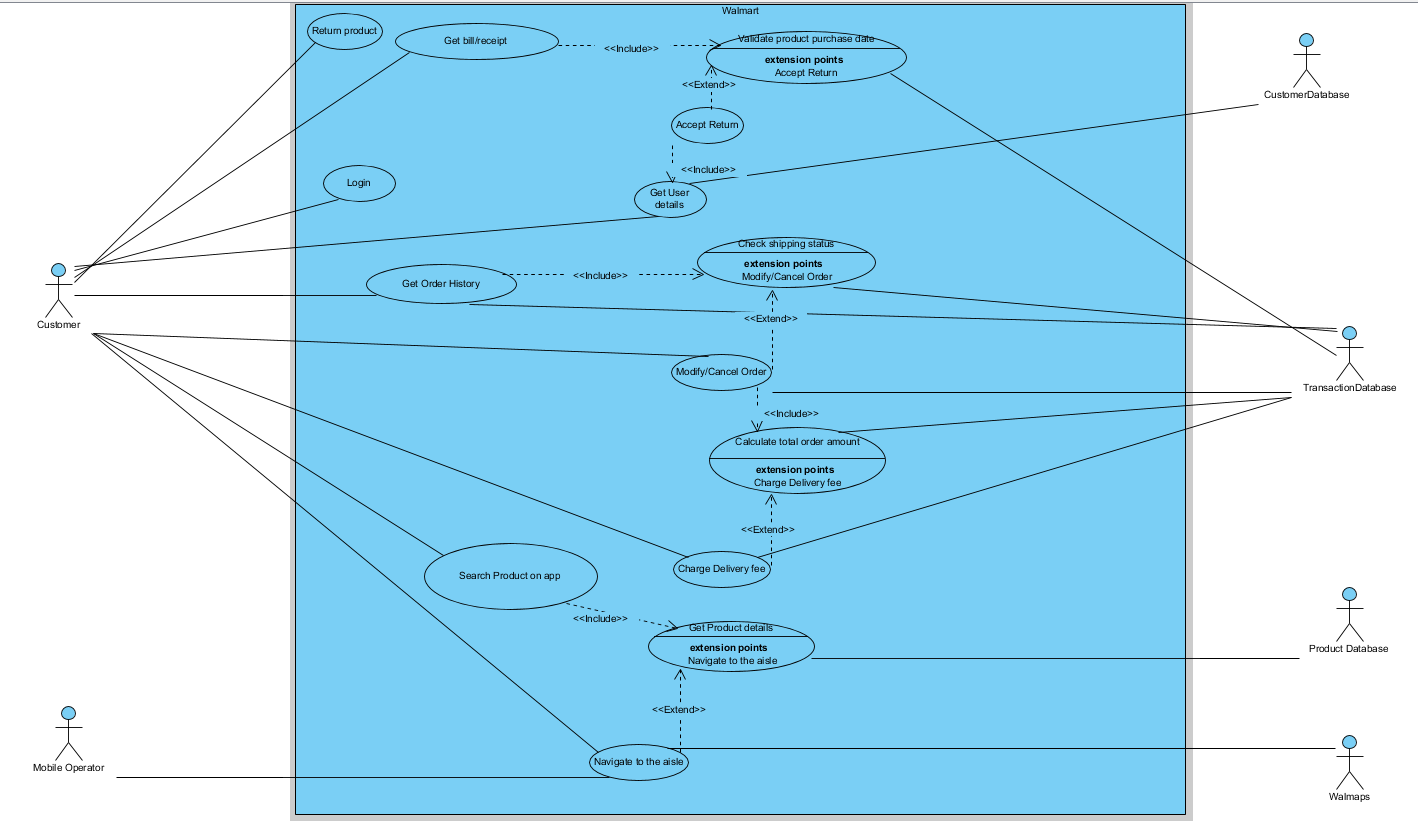
****

BPMN 3:

In Store Navigation – Choreography Diagram



USE CASE DIAGRAM:



USE CASE DESCRIPTION:

**Use Case Description 1:**

|  |
| --- |
| **Use Case Name:** Login |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** A customer will Login to view & modify his/her order |
| **Trigger:** When a customer clicks on Login button |
| **Normal flow of events:**  1. Customer navigates to Walmart website  2. Customer inputs Login Info i.e. username & password  3. Customer clicks on “Login” button on the main screen  4. Home page is displayed to the customer |
| **Exception:**  2a If a customer enters invalid login details, then display “Login failed” and repeat step 2 |

**Use Case Description 2:**

|  |
| --- |
| **Use Case Name:** Return Product |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** A customer will return the product at the Walmart store |
| **Trigger:** When a customer is unhappy with the product and do not want to use the product |
| **Normal flow of events:**  1. Customer uses the purchased product  2. Customer is unhappy with the product & wishes to return product  3. Customer visits the nearest Walmart store with the product |

**Use Case Description 3:**

|  |
| --- |
| **Use Case Name:** Get Bill/Receipt |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** A customer will provide the purchase receipt/bill |
| **Trigger:** When a Walmart Representative asks for a receipt/bill |
| **Normal flow of events:**  1. Customer is unhappy with the product & wishes to return the product  2. Walmart representative asks for product purchase receipt  3. Customer provides purchase receipt  4. Walmart representative checks the mode of transaction of the order |
| **Exception:**  3a. If a customer doesn’t have a receipt of purchase for the product, then ask for Customer Details |

**Use Case Description 4:**

|  |
| --- |
| **Use Case Name:** Get User Details |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** A customer will provide his details |
| **Trigger:** When Walmart representative validates & accepts the return, get customer details |
| **Normal flow of events:**  1. Walmart representative validates product for any damages  2. If product is validated, get customer details like Name, Email, Phone Number etc. |
| **Exception:**  1a. If product validation fails, do not accept the return. |

**Use Case Description 5:**

|  |
| --- |
| **Use Case Name:** Accept Return |
| **Primary Actor:** Walmart |
| **Stakeholder:** Customer |
| **Brief Description:** Walmart accepts return of the product and issue refund |
| **Trigger:** When the Purchased Product is validated |
| **Normal flow of events:**  1. Walmart validates the Purchased Product for 90 days purchase mark  2. Walmart checks the product for correctness  3. Walmart Accepts Return  4. Issue refund to the customer |
| **Exception:**  2a. If Product is not meeting the standards of correctness or purchase date, reject the item |

**Use Case Description 6:**

|  |
| --- |
| **Use Case Name:** Validate product purchase date |
| **Primary Actor:** Walmart |
| **Stakeholder:** Customer |
| **Brief Description:** Walmart rep views the receipt & validates the purchase date in the system |
| **Trigger:** When Walmart receives a bill/receipt |
| **Normal flow of events:**  1. Walmart receives a Receipt of the product  2. Walmart fetches the order history based on the receipt  3. Walmart checks the Purchased Product for the product in the system  4. Walmart validates the dates on receipt & system based on the purchase date which should be less than or equal to 90 days from the current date. |
| **Exception:**  3a. If product purchase date is greater than 90 days from the current date, reject the item |

**Use Case Description 7:**

|  |
| --- |
| **Use Case Name:** Get Order history |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** Customer views his/her order history from Walmart website |
| **Trigger:** When Customer clicks on “View Order History” button on website |
| **Normal flow of events:**  1. Customer clicks on “View Order History” button on the website  2. Customer’s order history is fetched from the database  3. Display the past purchased/canceled orders for the customer |
| **Exception:**  2a If there is a connection error between Walmart & its database, show “Try again later” error and repeat 3 |

**Use Case Description 8:**

|  |
| --- |
| **Use Case Name:** Check Shipping Status |
| **Primary Actor:** Walmart |
| **Stakeholder:** Walmart |
| **Brief Description:** Walmart fetches the shipping information |
| **Trigger:** When Customer clicks on “View Order History” button on website |
| **Normal flow of events:**  1. Customer clicks on “View Order History” button on the website  2. Shipping status of all the previous orders is fetched from the database |

**Use Case Description 9:**

|  |
| --- |
| **Use Case Name:** Modify/Cancel Order |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** Customer makes changes to his existing order |
| **Trigger:** When the customer clicks on “Cancel/Modify Order” button |
| **Normal flow of events:**  1. Customer is allowed to Cancel/Modify Order if the Shipping status is Not Shipped  2. Customer modifies or cancels the order by clicking on “Cancel/Modify Order” button  1.1 Modify order from the existing order  1.2 Customer cancels the entire order |
| **Exception flow of events:**  1. If the Shipping Status of previous orders is Shipped then don’t allow to Cancel/Modify Order |

**Use Case Description 10:**

|  |
| --- |
| **Use Case Name:** Calculate Total Order amount |
| **Primary Actor:** Transaction Database |
| **Stakeholder:** Walmart |
| **Brief Description:** Calculates total order amount for order which is Modified or Canceled |
| **Trigger:** When customer modifies or cancels the product from his/her order |
| **Normal flow of events:**  1. Customer modifies or cancels the order by clicking on “Cancel/Modify Order” button  1.1 Customer cancel/removes product(s) from the existing order  1.2 Customer cancels the entire order  2. Calculate the new Total Order Amount based on items present in the modified/canceled order |

**Use Case Description 11:**

|  |
| --- |
| **Use Case Name:** Charge Delivery fee |
| **Primary Actor:** Walmart |
| **Stakeholder:** Customer |
| **Brief Description:** Walmart levies delivery charges if the order amount is less than 35$ |
| **Trigger:** When customer modifies or cancels the product from his/her order |
| **Normal flow of events:**  1. Customer modifies the order by clicking on “Cancel/Modify Order” button  2. Customer removes product(s) from the existing order  3. Walmart calculates new Total Order Amount based on the items present in the modified/canceled order  4. If order amount is less than 35$ then add delivery charges to Total Order amount |
| **Exception:**  4a. If the order amount is greater than 35$, then exit |

**Use Case Description 12:**

|  |
| --- |
| **Use Case Name:** Search Product on app |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** Customer searches for products in Walmart Mobile Application |
| **Trigger:** When opens the Walmart Mobile Application |
| **Normal flow of events:**  1. Customer opens the Walmart Mobile Application  2. Customer types in Product Name in search box  3. Customer clicks on search button |
| **Exception:**  2a If Customer does not supply Product Name, show error and repeat 2 |

**Use Case Description 13:**

|  |
| --- |
| **Use Case Name:** Get Product details |
| **Primary Actor:** Product Database |
| **Stakeholder:** Walmart |
| **Brief Description:** Walmart accepts retrieves product details |
| **Trigger:** When customer clicks on “Search” button in mobile application with product name |
| **Normal flow of events:**  1. Customer opens the Walmart Mobile Application  2. Accept Product Name  3. Retrieve In-store product data along with its location from product database  4. Display product in mobile application |
| **Exception:**  3a If Product is not available in database, show error and repeat step 2 |

**Use Case Description 14:**

|  |
| --- |
| **Use Case Name:** Navigate to aisle |
| **Primary Actor:** Customer |
| **Stakeholder:** Walmart |
| **Brief Description:** In-store navigation to aisle/product location in Walmaps |
| **Trigger:** When customer clicks on “Navigate Me” button in Walmart app |
| **Normal flow of events:**  1. Walmart Mobile Application displays products along with location in mobile application  2. Customer clicks on Navigate Button in application which opens Walmaps  3. Walmaps gets customer location through mobile operator  4. Walmaps navigates customer to the corresponding aisle/shelf for the desired product |
| **Exception:**  3a If the customer’s location is not working, show error and repeat 2  3b If the customer’s location is not enabled, display enable location option |

**DATA DICTIONARY**

**Use Case**: Login

Login Info = Username + Password + CustomerID

User ID = Data Element

Password = Data Element

**Use Case Name:** Return Product

Product = Product Name + Product ID + Product Price + Product Type

Return Product = Return ID + Return Date + Reason + Return Status

Product Name = Data Element

Product ID = Data Element

Return Date = Data Element

Reason = Data Element

**Use Case Name:** Get Bill/Receipt

Receipt = Store Address + ReceiptID + Receipt Date

Customer Details = First Name + (Middle Name) + Last Name + ID Type+ ID Number + ID Expiration Date + Phone Number + Email ID

Product = Product Name + Product ID + Product Price + Product Type

Mode of Transaction = Payment Method + (Account Number) + (Expiration Date) + Signature + Transaction ID

Payment Method = [Check| Money Order| Gift Certificate| American Express| MasterCard| Visa| Other]

ID Type = [Govt|Private]

ID Number = Data Element

ID Expiration Date = Data Element

Phone Number = Data Element

Product Name = Data Element

Product ID = Data Element

Receipt Date = Data Element

Email ID = Data Element

**Use Case Name:** Get User Details

Customer Details = First Name + (Middle Name) + Last Name + ID Type+ ID Number + ID Expiration Date + Phone Number + Email ID

ID Type = [Govt|Private]

Email = Data Element

Phone Number = Data Element

ID Number = Data Element

ID Expiration Date = Data Element

Email ID = Data Element

**Use Case Name:** Accept Return

Product = Product Name + Product ID + Product Price + Product Type

Mode of Transaction = Payment Method + (Account Number) + (Expiration Date) + Signature + Transaction ID

Payment Method = [Check| Money Order| Gift Certificate| American Express| MasterCard| Visa| Other]

**Use Case Name:** Validate product purchase date

Receipt = Store Address + ReceiptID+ Receipt Date

Product = Product Name + Product ID + Product Price + Product Type

Order History = Order ID + Order Date + Order Amount + Order Quantity + Tax + Shipping

Shipping = Shipping ID + Address + Shipping Amount + Ship To Reason + Delivery Phone + Delivery Instruction + Shipping Status

Address = [In-store Address | Actual Address]

**Use Case Name:** Get Order history

Order History = Order ID + Order Date + Order Amount + Order Quantity + Tax + Shipping

Shipping = Shipping ID + Address + Shipping Amount + Ship To Reason + Delivery Phone + Delivery Instruction + Shipping Status

Address = [In-store Address | Actual Address]

**Use Case Name:** Check Shipping Status

Shipping status = [Shipped | Not Shipped]

**Use Case Name:** Modify/Cancel Order

Modify order = Modify Date + (Reason)

Shipping status = [Shipped | Not Shipped]

Modify Date = [Cancel Date | Removal Date]

**Use Case Name:** Calculate Total Order amount

Product = Product Name + Product ID + Product Price + Product Type

**Use Case Name:** Charge Delivery Fee

Product = Product Name + Product ID + Product Price + Product Type

**Use Case Name:** Search Product on app

Product = Product Name + Product ID + Product Price + Product Type

**Use Case Name:** Get Product details

In-store product data = Store Location + Product Aisle + Product Shelf + Latitude + Longitude

Product = Product Name + Product ID + Product Price + Product Type

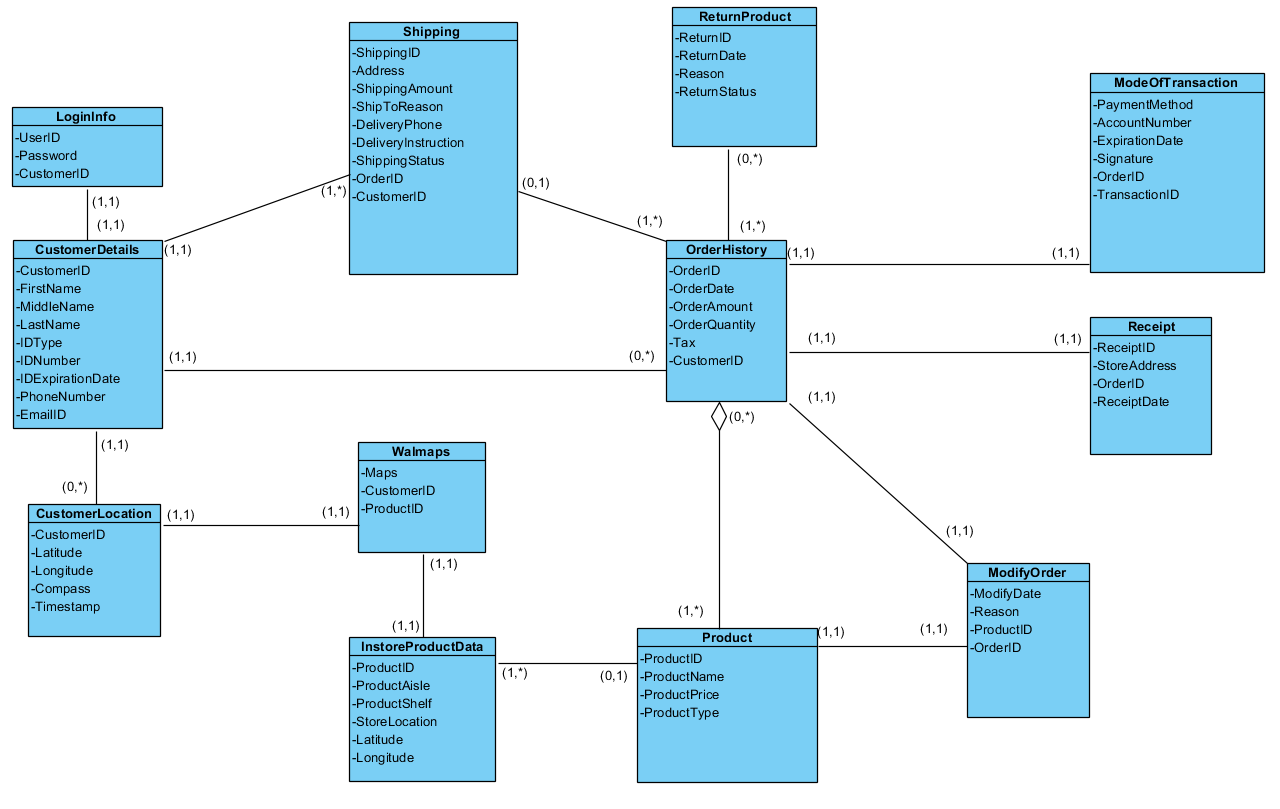
**Use Case Name:** Navigate to aisle

Customer location: Latitude + Longitude + Compass + Timestamp

Product = Product Name + Product ID + Product Price + Product Type

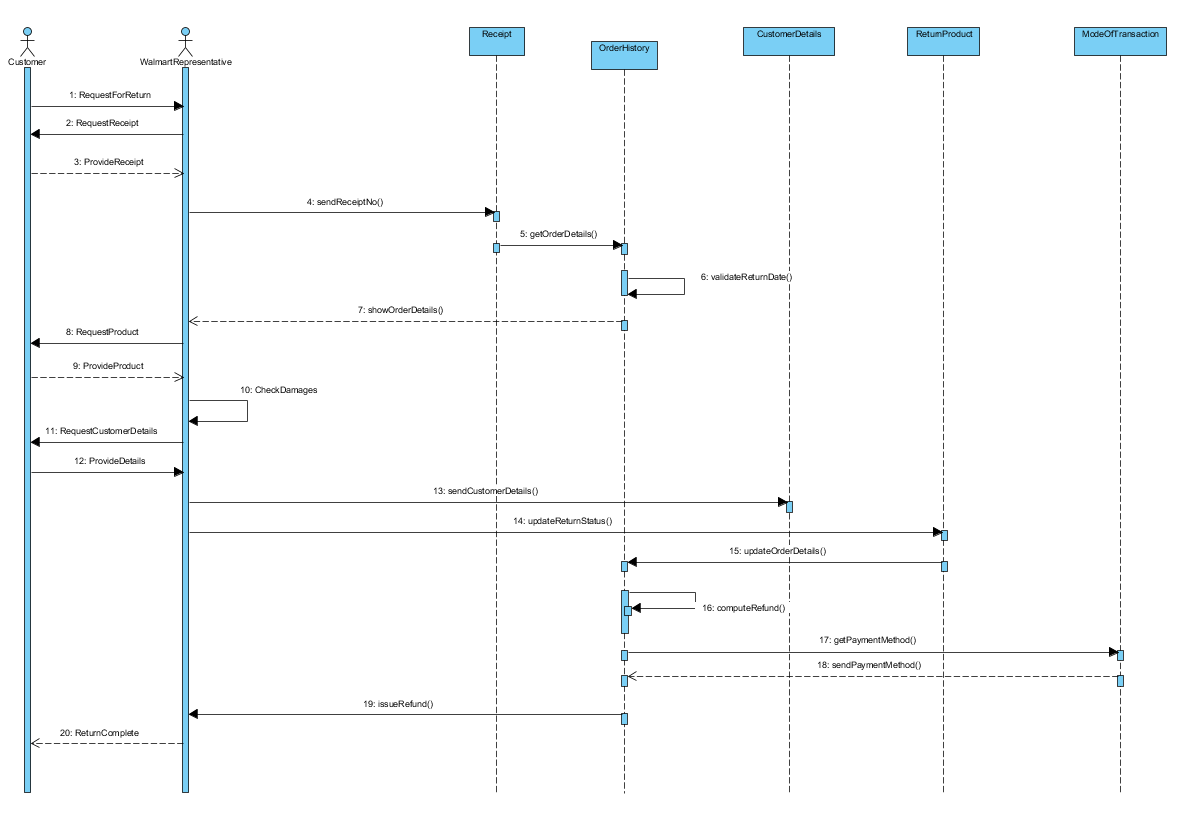
Walmaps = Maps + Product ID + Customer ID

**CLASS DIAGRAMS:**

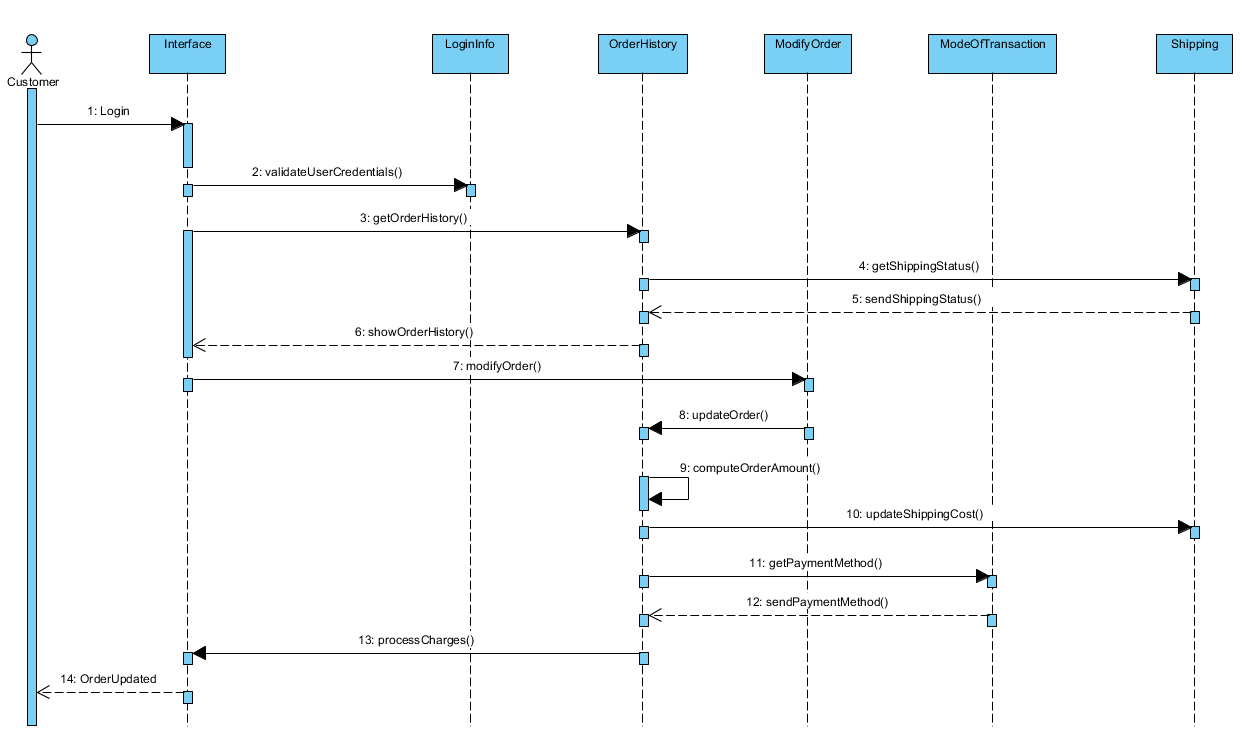


**SEQUENCE DIAGRAMS:**

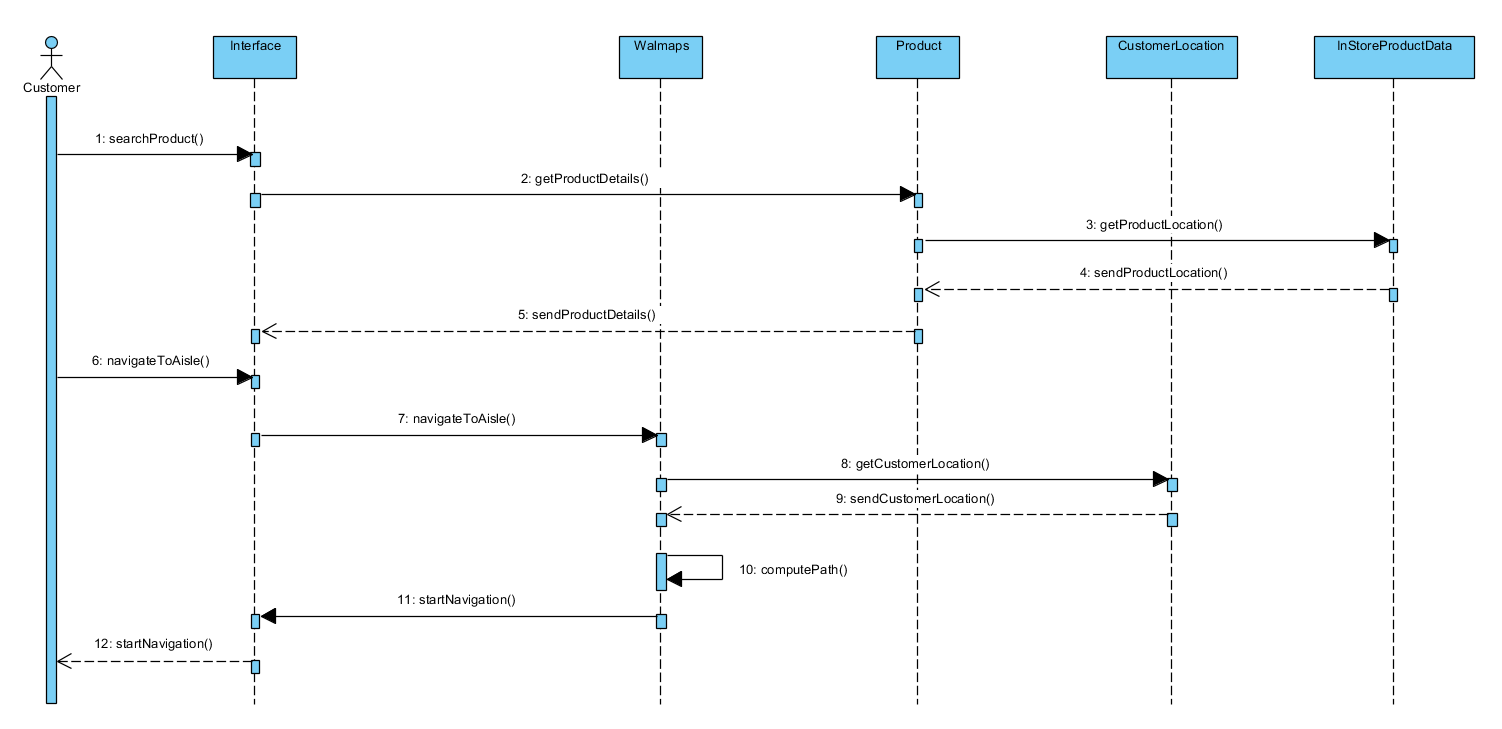
Return



Online order

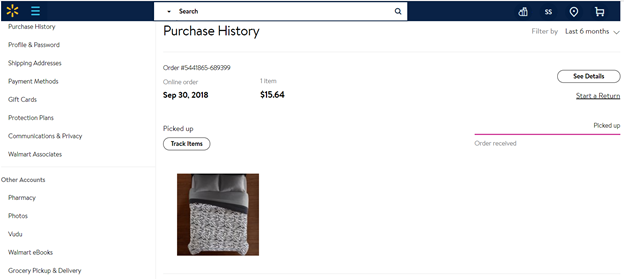


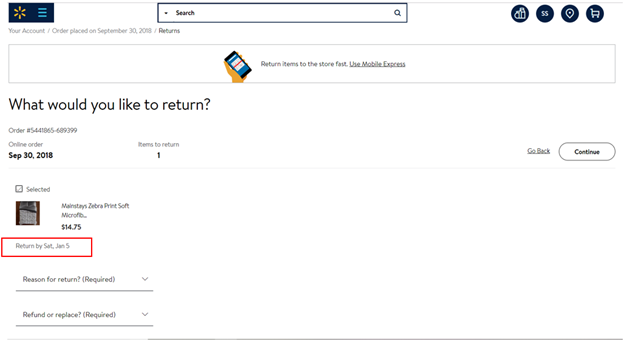
Navigation



INTERFACE DESIGN

90 Days Return Policy misuse:





DATABASE DESIGN:

Table: LoginInfo  
(*CustomerID*, Username, Password)  
**Constraints**  
Primary Key: CustomerID, must be UNIQUE and NOT NULL

Foreign Key: CustomerID  
Username & Password should be NOT NULL

Table: CustomerDetails  
(CustomerID, FirstName, MiddleName, LastName, IDType, IDNumber, IDExpirationDate, PhoneNumber, EmailID)  
**Constraints**  
Primary Key: CustomerID, must be UNIQUE and NOT NULL

Table: CustomerLocation  
(*CustomerID*, Latitude, Longitude, Compass, Timestamp)  
  
**Constraints**  
Primary Key: CustomerID must be UNIQUE and NOT NULL  
Foreign Key: CustomerID

Table: Walmaps  
(*CustomerID*, *ProductID,* Maps)  
  
**Constraints**  
Primary Key: CustomerID & ProductID must be UNIQUE and NOT NULL  
Foreign Key: CustomerID & ProductID

Table: Shipping  
(ShippingID, *OrderID*, *CustomerID*, Address, ShippingAmount, ShipToReason, DeliveryPhone, DeliveryInstruction, DeliveryStatus)  
  
**Constraints**  
Primary Key: ShippingID must be UNIQUE and NOT NULL  
Foreign Key: OrderID, CustomerID

Table: OrderHistory  
(OrderID, *CustomerID*, OrderDate, OrderAmount, OrderQuantity, Tax )  
  
**Constraints**  
Primary Key: OrderID must be UNIQUE and NOT NULL  
Foreign Key: CustomerID

Table: ReturnProduct  
(ReturnID, ReturnDate, Reason, ReturnStatus )  
  
**Constraints**  
Primary Key: ReturnID must be UNIQUE and NOT NULL

Table: OrderReturn  
(*ReturnID, OrderID* )  
  
**Constraints**  
Primary Key: ReturnID & OrderID must be UNIQUE and NOT NULL  
Foreign Key: ReturnID & OrderID

Table: OrderReturn  
(*ReturnID, OrderID* )

(Note: This is an association table between OrderHistory & ReturnProduct to eliminate many to many relationship)

**Constraints**  
Primary Key: ReturnID & OrderID must be UNIQUE and NOT NULL  
Foreign Key: ReturnID & OrderID

Table: Product  
(ProductID, ProductName, ProductPrice, ProductType )  
  
**Constraints**  
Primary Key: ProductID must be UNIQUE and NOT NULL

Table: OrderProduct  
(*ProductID, OrderID* )  
(Note: This is an association table between OrderHistory & Product to eliminate many to many relationship)

**Constraints**  
Primary Key: ProductID & OrderID must be UNIQUE and NOT NULL  
Foreign Key: ProductID & OrderID

Table: InstoreProductData  
(*ProductID*, ProductAisle, ProductShelf, StoreLocation, Latitude, Longitude )  
  
**Constraints**

Primary Key: ProductID must be UNIQUE and NOT NULL  
Foreign Key: ProductID

Table: ModifyOrder  
(*ProductID*, *OrderID*,  ModifyDate, Reason )  
  
**Constraints**

Primary Key: ProductID & OrderID must be UNIQUE and NOT NULL

Foreign Key: ProductID & OrderID

Table: ModeOfTransaction  
(TransactionID, *OrderID,* PaymentMethod, AccountNumber, ExpirationDate, Signature )  
  
**Constraints**  
Primary Key: TransactionID must be UNIQUE and NOT NULL  
Foreign Key: OrderID

Table: Receipt  
(ReceiptID, *OrderID,* ReceiptDate, StoreAddress )  
  
**Constraints**  
Primary Key: ReceiptID must be UNIQUE and NOT NULL  
Foreign Key: OrderID

COMPLETE CLASS DIAGRAMS WITH FUNCTIONS:

