URBAN WRAPPER

BTM 495-CC
Team 3
Final Report
Presented to:
Mr. Hossein Azarpanah

John Molson School of Business
Monday, August 16th, 2021

Prepared by:
Clive Dency Nya
Chaimaa Nazrati
Jonathan Tartaglia
Benjamin Groot
Faisal Alshowaeir
Mariam Traore
Estefania Castillo

Table of Contents

Executive Summary

Urban Wrapper is a clothing and accessories company founded in 2018 by Mrs. Camina Harrison-Chéry. The Montreal business offers handmade headwraps, headbands, scrunchies and durags made of African wax fabric. In this report we will investigate the company and illustrate the proposed solutions that came about after an in-depth analysis of its current structure and processes.

To understand the challenges faced by Urban Wrapper we need to understand who they are, and what is their objective. The goal for Urban Wrapper is to bring awareness of the African culture by providing accessories for both male and females of all origins. However, the business strategy still has a lot of room for growth as the company was founded only three years ago as a passion project. As for Urban Wrapper's IT strategy, little resources were spent on developing it in the past years. They currently have two platforms: the official Urban Wrapper website and a growing presence on the mobile app, Instagram.

Urban Wrapper has two business functions/subsystems: the procurement and events sales. On a high level, the activities in the procurement subsystem involve contacting the supplier, inspecting the fabric by touch, purchasing it, allocating the portions for her seamstresses and herself, then finally, updating her notes and the website. The processes in the event sales include getting knowledge of events, selecting potential events, registering to the event, and selling at the event.

However, because of the minimum focus that were attributed to its IT system, Urban Wrapper has experienced many problems that could be brought down to 3 main business problems which are.

- No uniformity in data collection which led to data loss, and data not been accessed in a timely manner. Data is stored in many different formats such as on phone apps, a laptop, the website, pieces of paper, and most often in her own mind. This is caused by the lack of a proper IT infrastructure.
- Many processes are done manually. Because of this, a lot of time is wasted, and in turn affects the company's profitability. Mrs. Harrison-Chéry also has several business needs and one requirement.
- No clear processes or documentation for steps to follow. She needs a method to track how much inventory is left over at events and a method to track how much raw material is bought and stored. To do so, the requirement is to implement an automated stock control and inventory tracking management system.

Team Project Proposal

Knowledge Area

Each team member had a different background and area of expertise which was a strength for the success of the project. The table below shows a breakdown of the knowledge area with its lead and support team member.

Knowledge Area	Lead Team Member	Support Team Member
KA1 - Information Systems Analysis and Design	Jonathan Tartaglia, Mariam	Faisal Alshowaier
KA2 - Database Management	Chaimaa Nazrati	Mariam Henriette Traoré
KA3 - Human-Computer Interaction (HCI - UI/UX)	Estefania Castillo, Clive Dency	Benjamin Groot
KA4 - Programming and Development	Benjamin Groot	Jonathan Tartaglia, Estefania Castillo
KA5 - Project Management	Faisal Alshowaier	Chaimaa Nazrati, Clive Dency

BTM481 Project Information

481 Project	Name of the project
481 Project Details	All details such as the narratives, proposed DFD's and use cases will be found in appendix 7, 8 and 9.
Semester of Project Completion	Fall 2020

Professor of 481 section	Dr. Meral Buyukkurt
Status of 481 Client:	How was the previous experience with the client? = We had an excellent experience with Camina. She was present in every meeting, answered our questions and was happy with our recommendations. Currently, have you been able to reach and confirm with the client? - Yes How likely are they to continue their participation? - Very likely, she has good communication and is very easy to reach.

495 Project Update	
Project Sponsor:	Camina Harrison-Chéry
Business Need:	- Optimize the owner's process of keeping track of fabric availabilities from different suppliers. - Document the amount of raw material bought, stored, and shared with Urban Wrapper's two seamstresses and Camina herself. - Create a database to track her inventory and customer's information for events.
Business Requirements:	Business requirements highlighted the criteria that needed to be met to solve the identified problems. Urban Wrapper encountered data loss and had trouble accessing data remotely and efficiently, which consequently affected the business overall efficacy. The team identified the following requirements where the mandatory criteria are to be met in order to maximize Urban Wrappers' success:

	1. Automizing manual work 2. Automating inventory management 3. Optimizing sales at events
Business Value:	Expected value that the system will provide: - Prevent any redundant processes and keeping track of sales - Optimization of Inventory management that is linked with sales of the merchandise on hand
Special Issues or Constraints:	2 main constraints/special issues were linked to the project. - Budget - The business owner is not tech savvy, so she needs an application with a good user interface

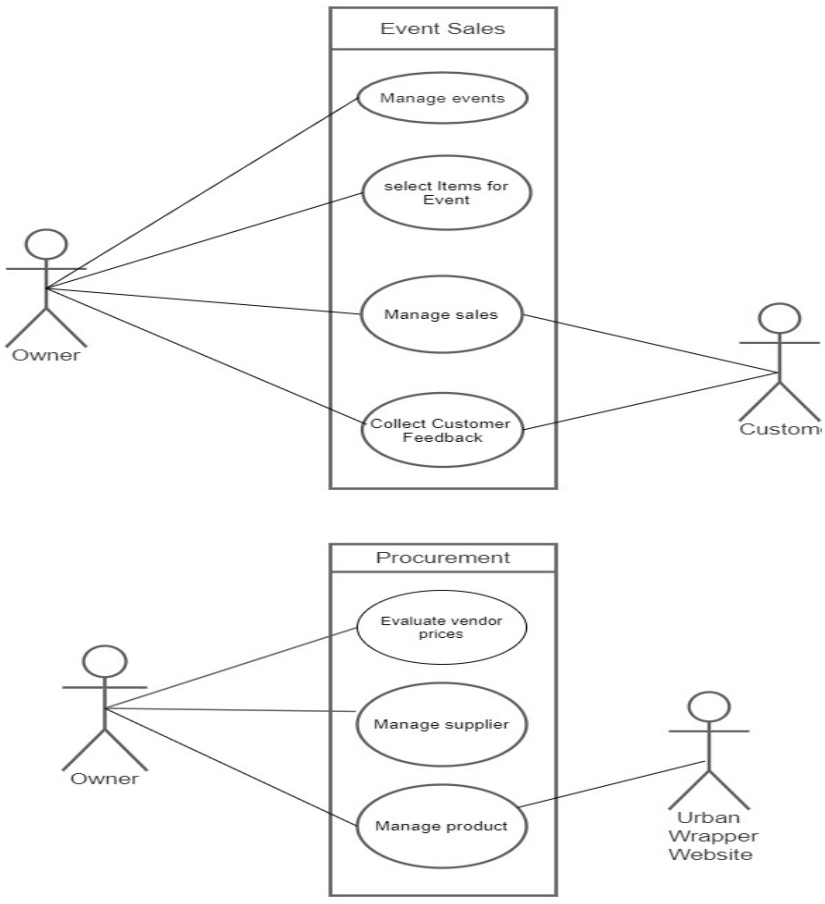
Functional Modeling

Use Case Diagram

Notes on revisions made to the use case diagram

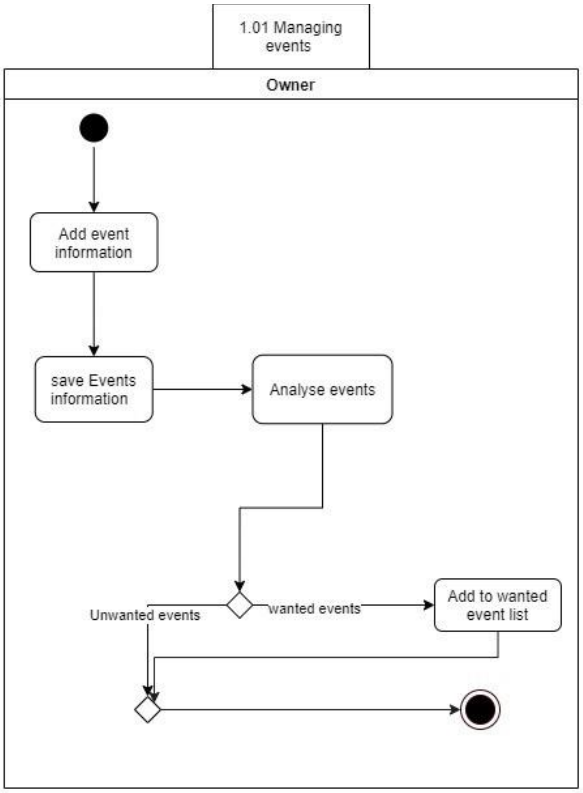
Below is a representation of our latest use case diagram. When comparing to our old version (found in appendix 4), we can note the following changes.

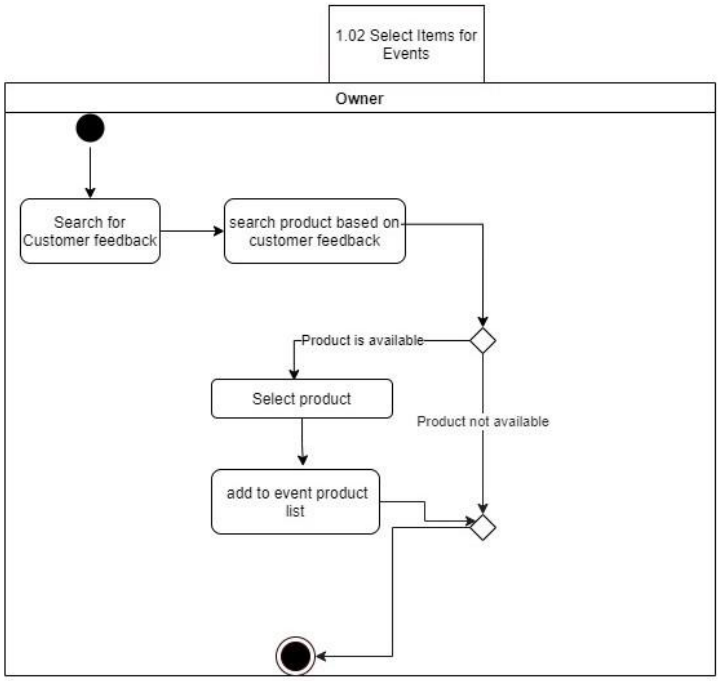
- We removed the following use cases: register to the event, make payment, collect customer details for feedback, contact customer for feedback, acquire product. After the professor's comments, we realized that these use cases were unnecessary.
- We changed the names of some of our use cases because we found that some names didn't make sense, and some of them could be merged.

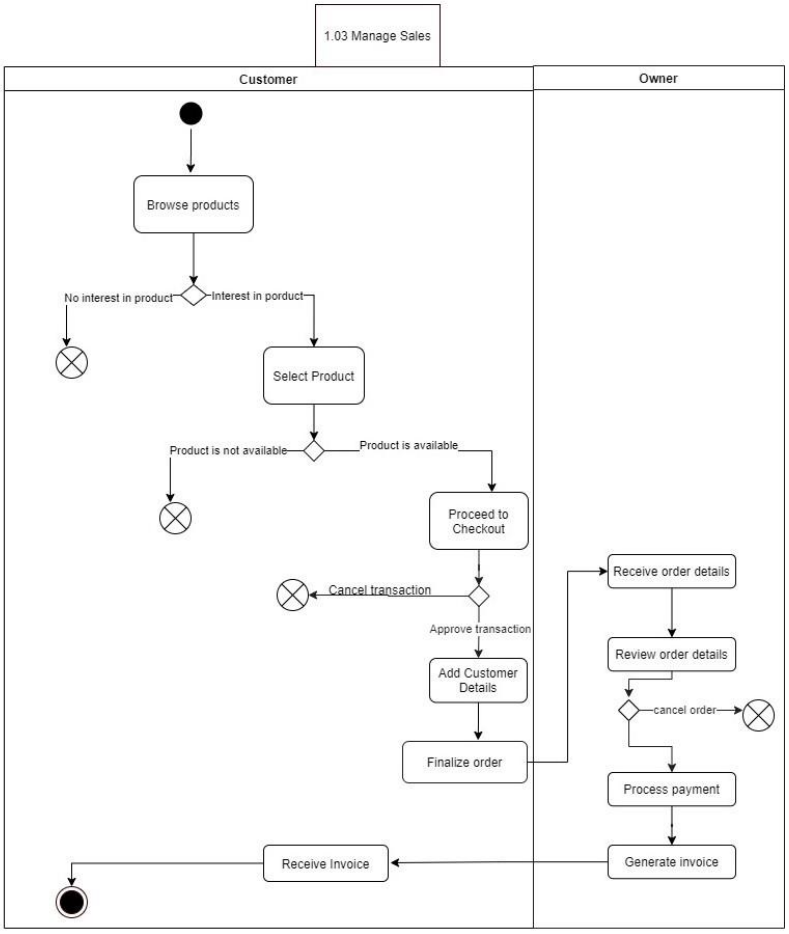


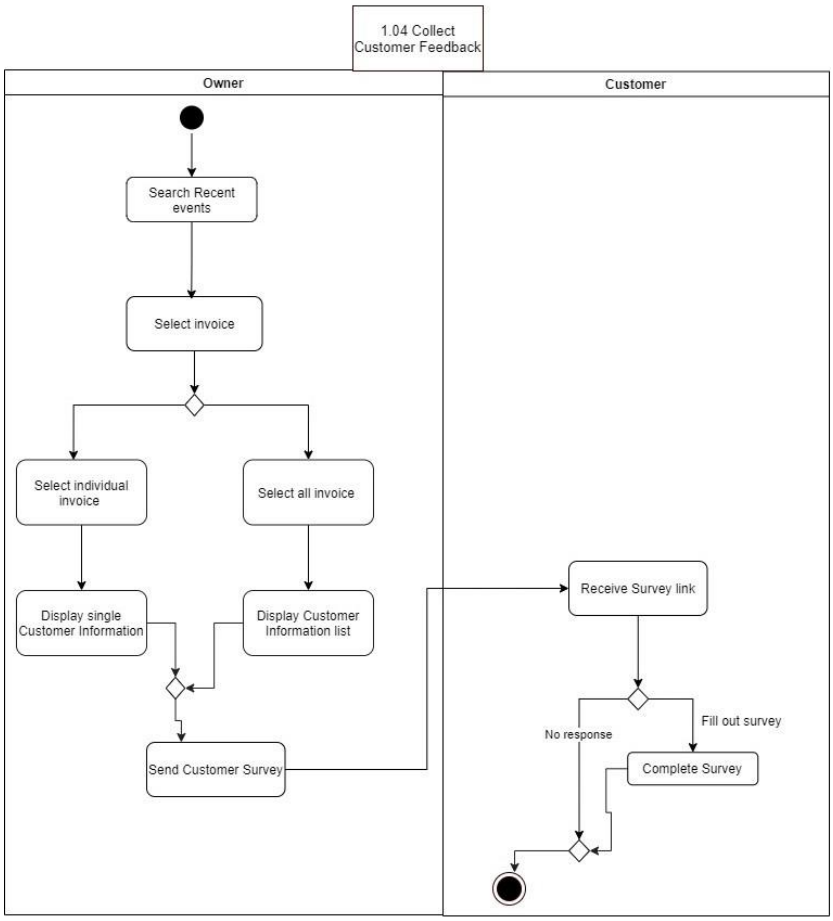
Activity Diagram

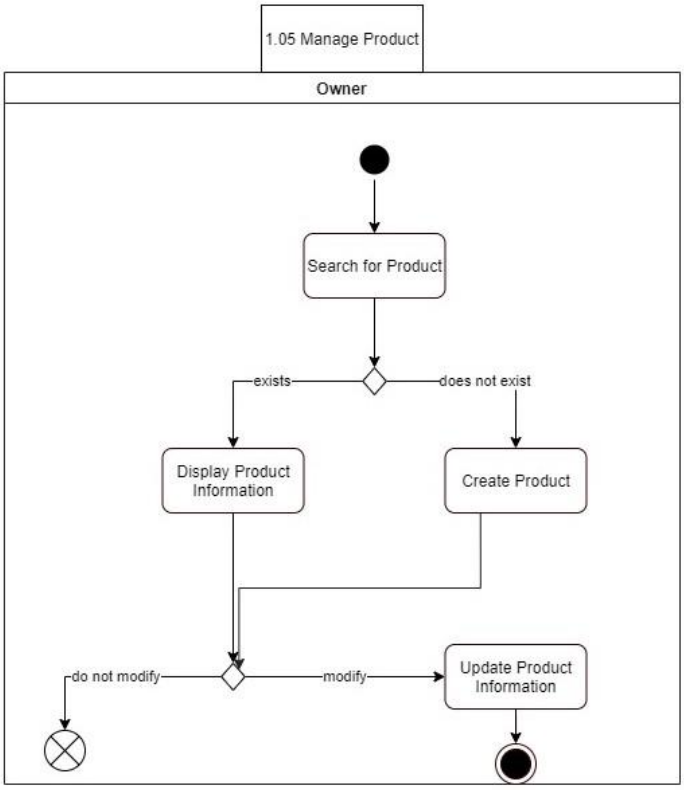
After modifying the use case diagram, we made modifications to our activity diagram to match it with our revised use case diagram. Moreover, we realized that some activities were too broad; therefore, we detailed them more.

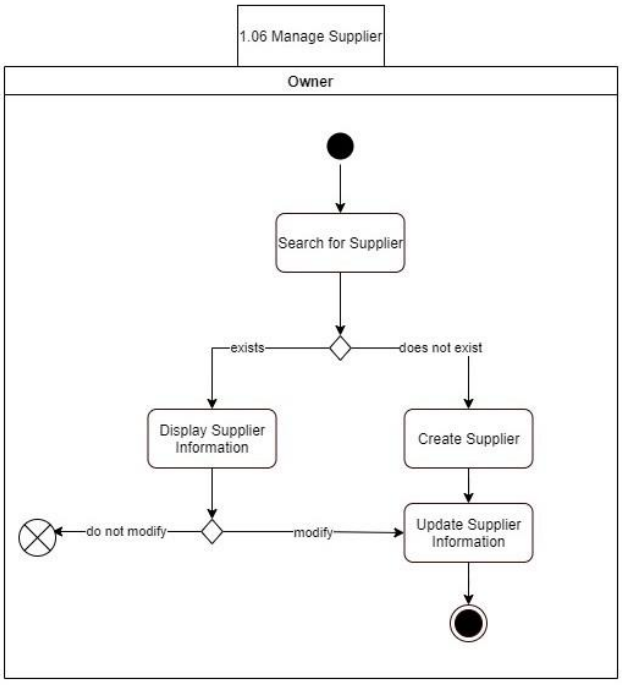


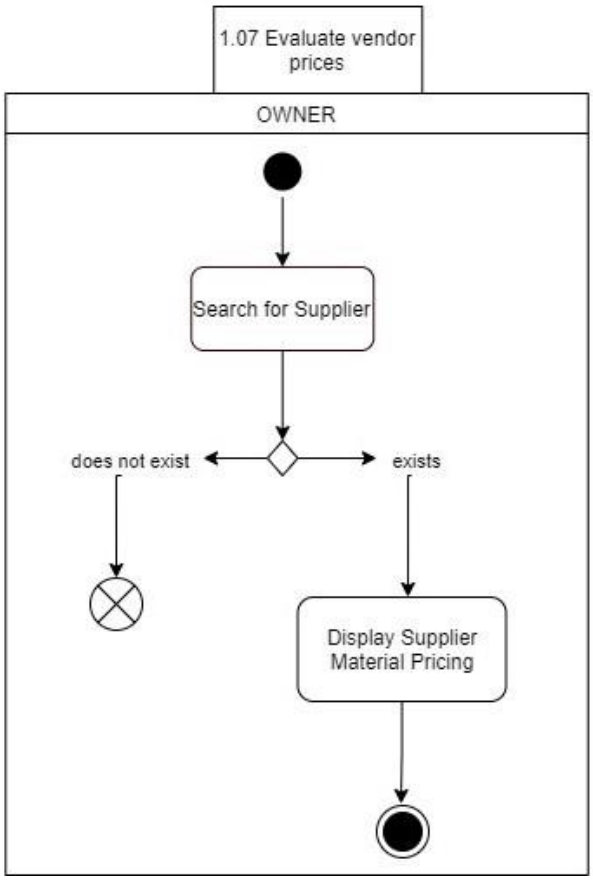

1.02 Select Items for Events
Owner
Search for Customer feedback
search product based on customer feedback
Product is available
Select product
Product not available
add to event product list


1.03 Manage Sales
Customer
Owner
Browse products
No interest in product
Interest in porduct
Select Product
Product is not available
Product is available
Proceed to Checkout
Cancel transaction
Approve transaction
Add Customer Details
Finalize order
Receive order details
Review order details
cancel order
Process payment
Generate invoice
Receive Invoice


1.04 Collect Customer Feedback
Owner
Customer
Search Recent events
Select invoice
Select individual invoice
Select all invoice
Display single Customer Information
Display Customer Information list
Send Customer Survey
Receive Survey link
No response
Fill out survey
Complete Survey


1.05 Manage Product
Owner
Search for Product
exists
does not exist
Display Product Information
Create Product
do not modify
modify
Update Product Information


1.06 Manage Supplier
Owner
Search for Supplier
exists
does not exist
Display Supplier Information
Create Supplier
do not modify
modify
Update Supplier Information


1.07 Evaluate vendor prices
OWNER
Search for Supplier
does not exist
exists
Display Supplier Material Pricing

Structural Modelling

The goal of structural modeling is to illustrate the system's objects and their relationships, this includes all the people, places, and anything that influences the system such as events for example. Below are the classes defined in the scope of our project along with a brief explanation of their role when interacting with the system.

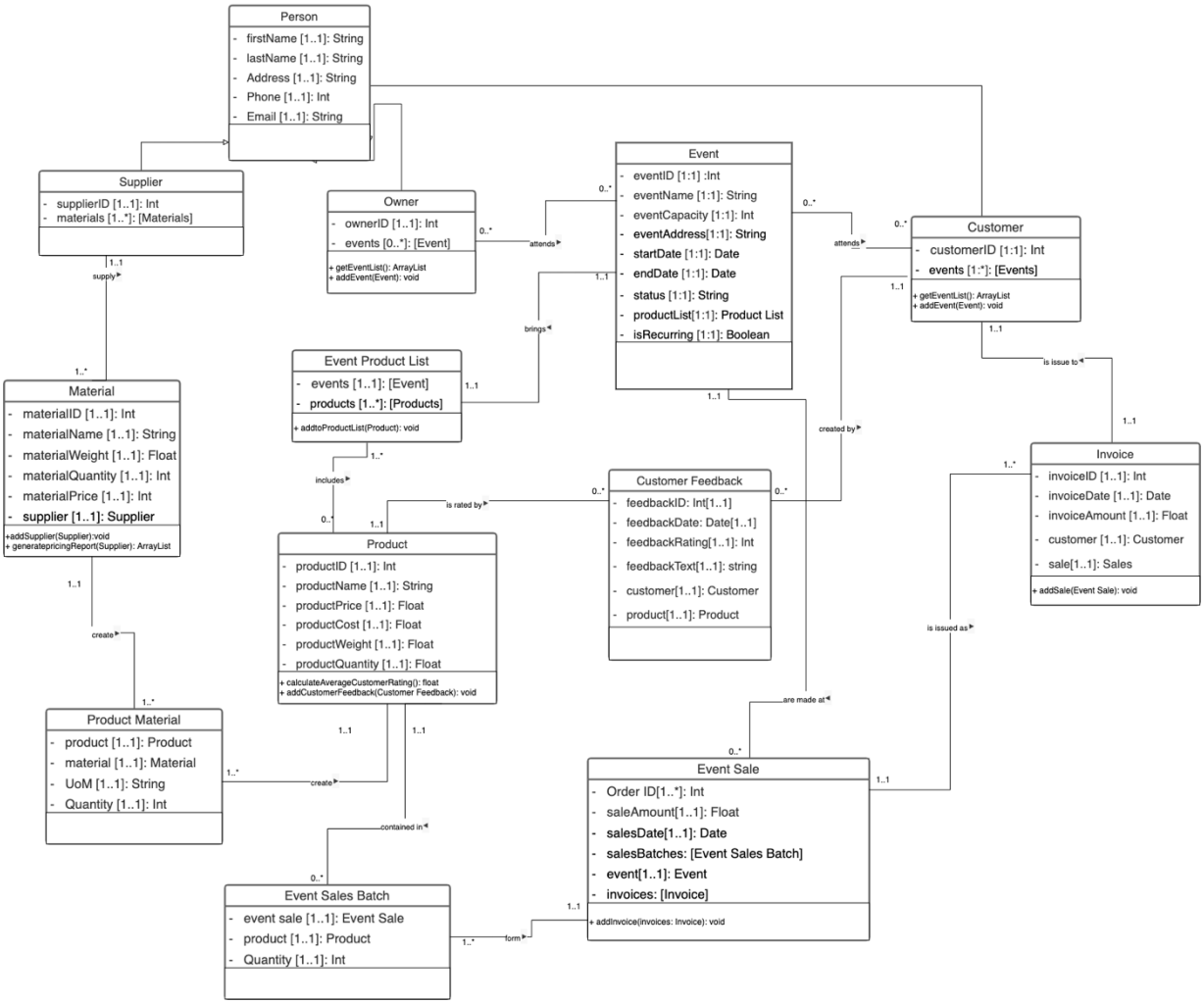
- Customer: A customer of urban Wrapper is one who comes to the event, makes an order, then pays. When interacting with the system, the customer browses the items he wants to buy, selects the items, and adds them to an item list (Check out cart). Once that is done, he records his information (as shown in the attributes name, address, etc). After the sale is made, he receives a link for the feedback, which he can complete, and send back.
- Owner: The owner here is Camina Urban Wrapper founder. She is the main entity that takes the actions of adding, updating, and modifying information with full access to every part of the system. With that access, she can register the list of products that will be brought to the event. She processes payment equally once the customer confirms the order.
- Event product list: This is the list of products that the owner will sell in the events based on the feedback she receives from customers.
- Events: An event represents an occurrence where the owner attends to showcase and sell her products. In the database every event needs to include a date, an address, capacity, and more information that will help to properly analyze it and see if it meets the criteria.
- Events sale: Event sales are the sales made at the event. A sale is made when the customer choses his product and adds his information to the system. Once that is done, he confirms the order, there we consider a sale is made.
- Product: The product is what is being sold at the event.
- Invoices: Invoices are proof that a transaction was completed. After the sale is done, the owner processes payment, and a receipt is sent to the Customer email.
- Customer feedback: Customer feedback is the review that the customer will attribute to the product after the purchase.
- Suppliers: A supplier is the entity responsible for selling/providing fabrics or materials to produce finished goods. The supplier doesn't interact with the system, but records of what he sells, and his prices are stored in the system
- Material: Materials represent the fabrics needed to produce the finished products. (Headbands, headwraps etc.)
- Event Sales Batch: Event sales batch is the class that contains all the sales made during events. It contains info such as the type of products sold and their quantities.
- Product Material: Product material represents the "recipe", or the combinations of fabrics needed to create each product.

Notes on revisions made to the class diagram

Below is a representation of our latest class diagram. When comparing to our old version (found in appendix 6), we can note the following changes.

- The removal of methods such as get customer, constructor, and destructor (Example under events class). This is because these methods are logical functions that are known to exist with this class. It wasn't necessary to show a method that logically needs to exist with that class.
- The addition of the following 4 classes; Event product list, customer feedback, product material, and event sales batch. Even though each class has a different role and purpose as explained earlier, the reason they were added is the same in general. Our model couldn't be complete if they were not added. Each additional class brought an extra layer of detail to our model.
- Changed the name for Sales ID to Order ID. This change was made because as we started drawing the sequence diagram, and building the prototype, we realized that we didn't have an order class, and what we referred to as a sale was an order. Our definition of a sale became essentially the same as that of an order because in our definition, a sale doesn't need a payment to be completed. So, we changed the name to order ID because we already had an order ID that would be generated when a customer chooses his product and adds his information to the system.

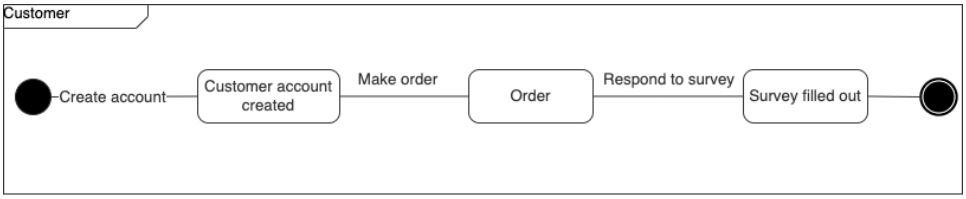
1

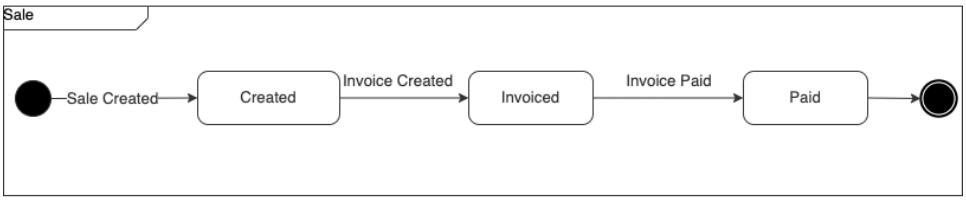

Person
- firstName [1..1]: String
- lastName [1..1]: String
- Address [1..1]: String
- Phone [1..1]: Int
- Email [1..1]: String
Supplier
- supplierID [1..1]: Int
- materials [1..*]: [Materials]
Owner
- ownerID [1..1]: Int
- events [0..*]: [Event]
+ getEventList(): ArrayList
+ addEvent(Event): void
Event
- eventID [1:1] :Int
- eventName [1:1]: String
- eventCapacity [1:1]: Int
- eventAddress[1:1]: String
- startDate [1:1]: Date
- endDate [1:1]: Date
- status [1:1]: String
- productList[1:1]: Product List
- isRecurring [1:1]: Boolean
Customer
- customerID [1:1]: Int
- events [1:*]: [Events]
+ getEventList(): ArrayList
+ addEvent(Event): void
Material
- materialID [1..1]: Int
- materialName [1..1]: String
- materialWeight [1..1]: Float
- materialQuantity [1..1]: Int
- materialPrice [1..1]: Int
- supplier [1..1]: Supplier
+addSupplier(Supplier):void
+ generatepricingReport(Supplier): ArrayList
Event Product List
- events [1..1]: [Event]
- products [1..*]: [Products]
+ addtoProductList(Product): void
Customer Feedback
- feedbackID: Int[1..1]
- feedbackDate: Date[1..1]
- feedbackRating[1..1]: Int
- feedbackText[1..1]: string
- customer[1..1]: Customer
- product[1..1]: Product
Invoice
- invoiceID [1..1]: Int
- invoiceDate [1..1]: Date
- invoiceAmount [1..1]: Float
- customer [1..1]: Customer
- sale[1..1]: Sales
+ addSale(Event Sale): void
Product
- productID [1..1]: Int
- productName [1..1]: String
- productPrice [1..1]: Float
- productCost [1..1]: Float
- productWeight [1..1]: Float
- productQuantity [1..1]: Float
+ calculateAverageCustomerRating(): float
+ addCustomerFeedback(Customer Feedback): void
Product Material
- product [1..1]: Product
- material [1..1]: Material
- UoM [1..1]: String
- Quantity [1..1]: Int
Event Sale
- Order ID[1..*]: Int
- saleAmount[1..1]: Float
- salesDate[1..1]: Date
- salesBatches: [Event Sales Batch]
- event[1..1]: Event
- invoices: [Invoice]
+ addInvoice(invoices: Invoice): void
Event Sales Batch
- event sale [1..1]: Event Sale
- product [1..1]: Product
- Quantity [1..1]: Int
supply
attends
attends
brings
includes
is rated by
created by
is issue to
create
create
contained in
are made at
is issued as
form

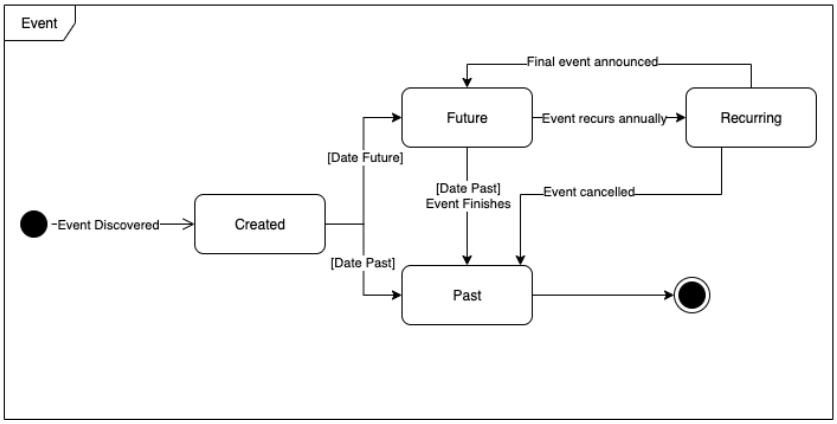
Behavioral Modelling

The following modifications were made to the behavior modeling:

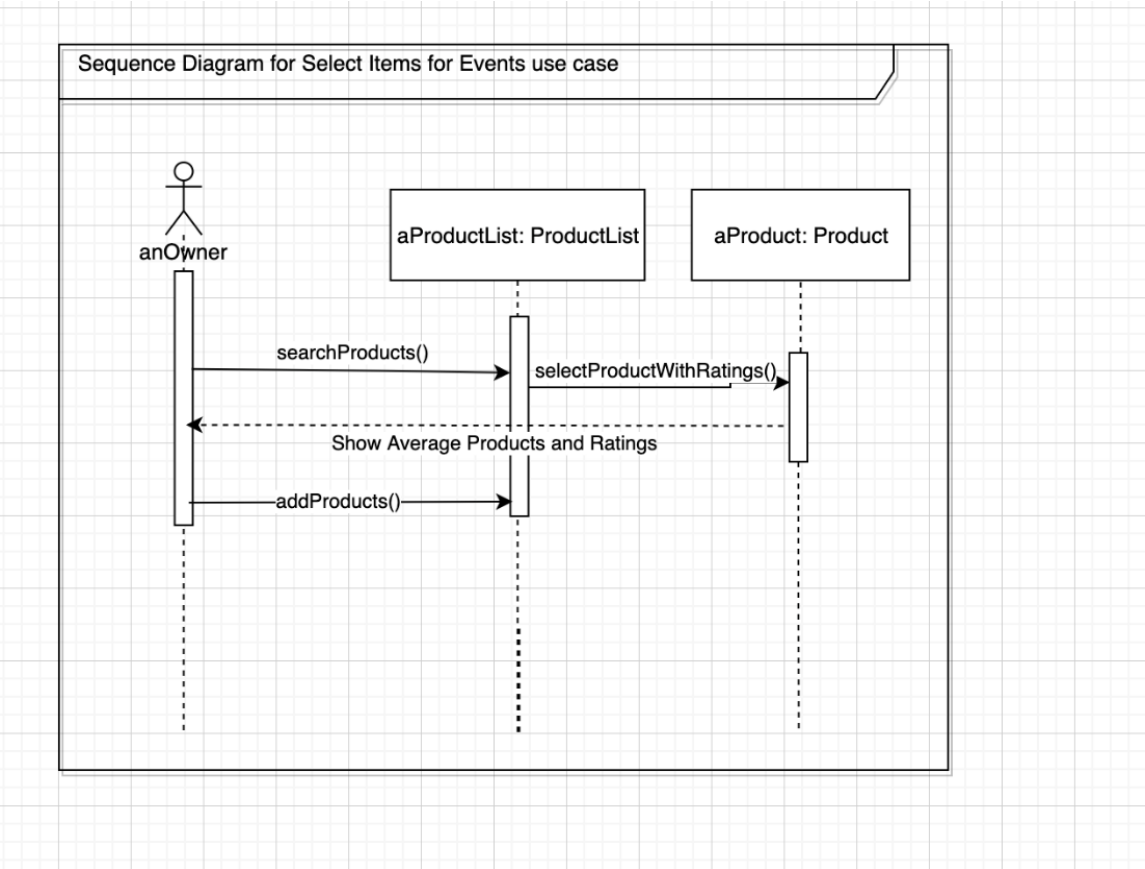
- We needed to make a modification by removing Customer and replaced it with the sale Behavioral modelling as simplified the flow of information.
- The modification was made to create a better flow of information from the time the sale is created until the order it complete paid.
- No modification was needed for the Event.







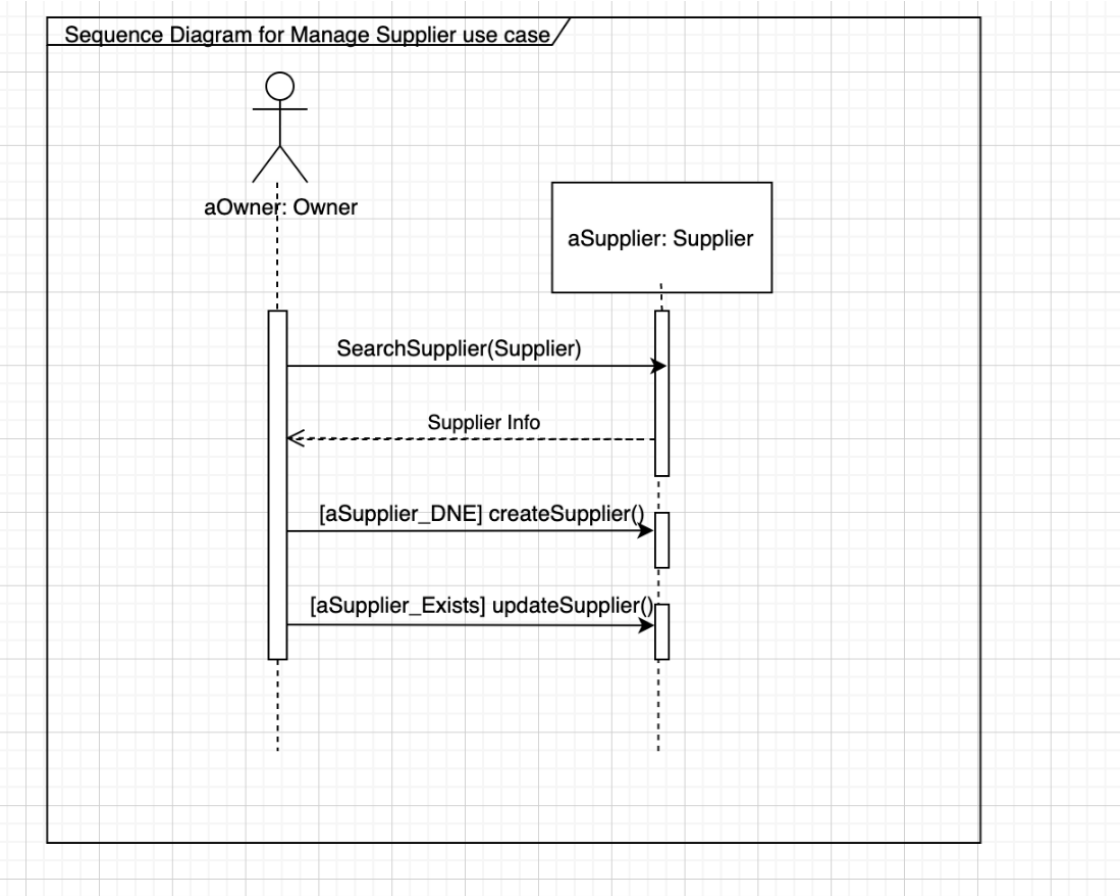
Select items for events Sequence diagram



Some changes were made to the Select Items for Events use case;

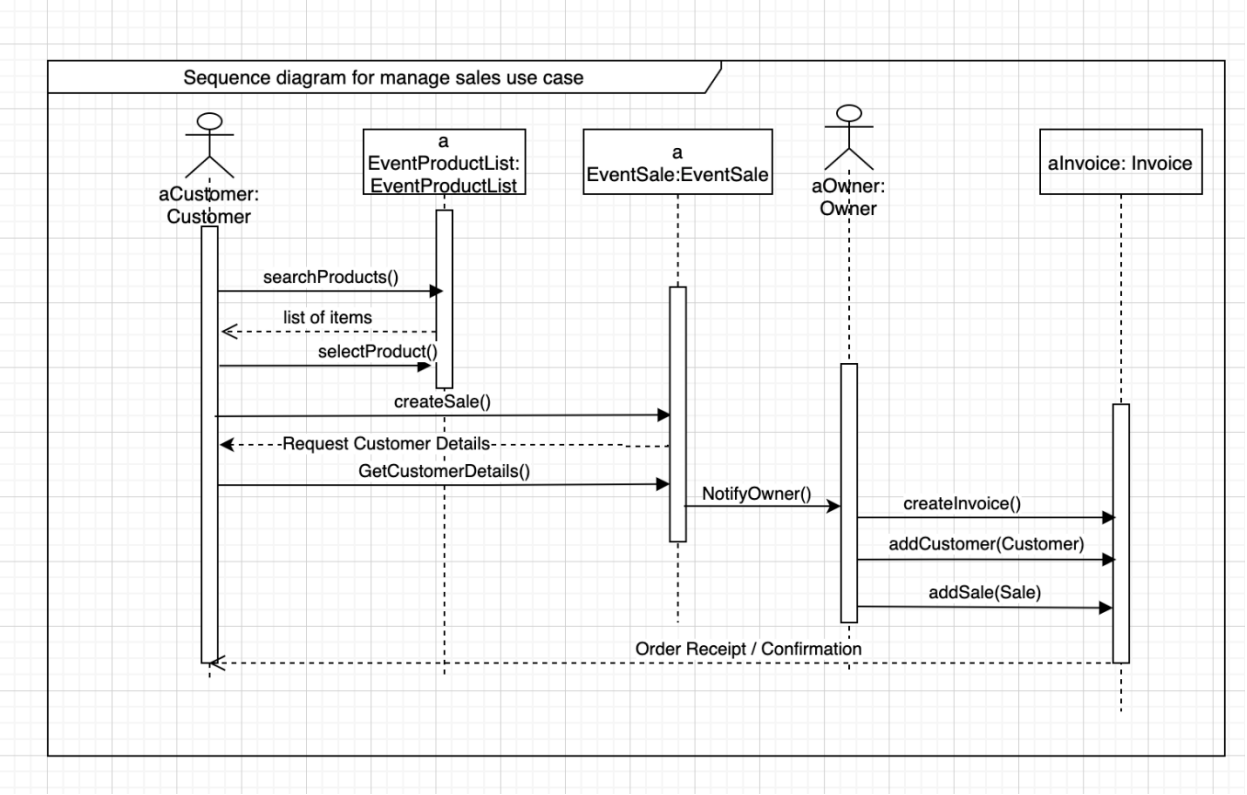
- A new class was added aProduct:ProductList
- All outflow and inflows of information were modified
- "Get aggregate results()", "Show survey Aggreagate Results", "selectProducts", "addProduct(Int)", "request Product Quantity" and addProductQuantity() were removed.
- The adjusted information were made was the inflow information for owner to ProductList "SearchProducts()", ProductList to Porduct add an inflow of information selectProductWith Ratings().
- Outlfow of information from Product back to Owner was added "Show Average Products and Ratings"
- Inflow of information was added from Owner to ProductsList "addProducts()"

Manage Supplier Sequence diagram



- In this sequence diagram, we removed the “getsupplierID” outflow because it did not need information from the “createSupplier()” to “updateSupplier()”

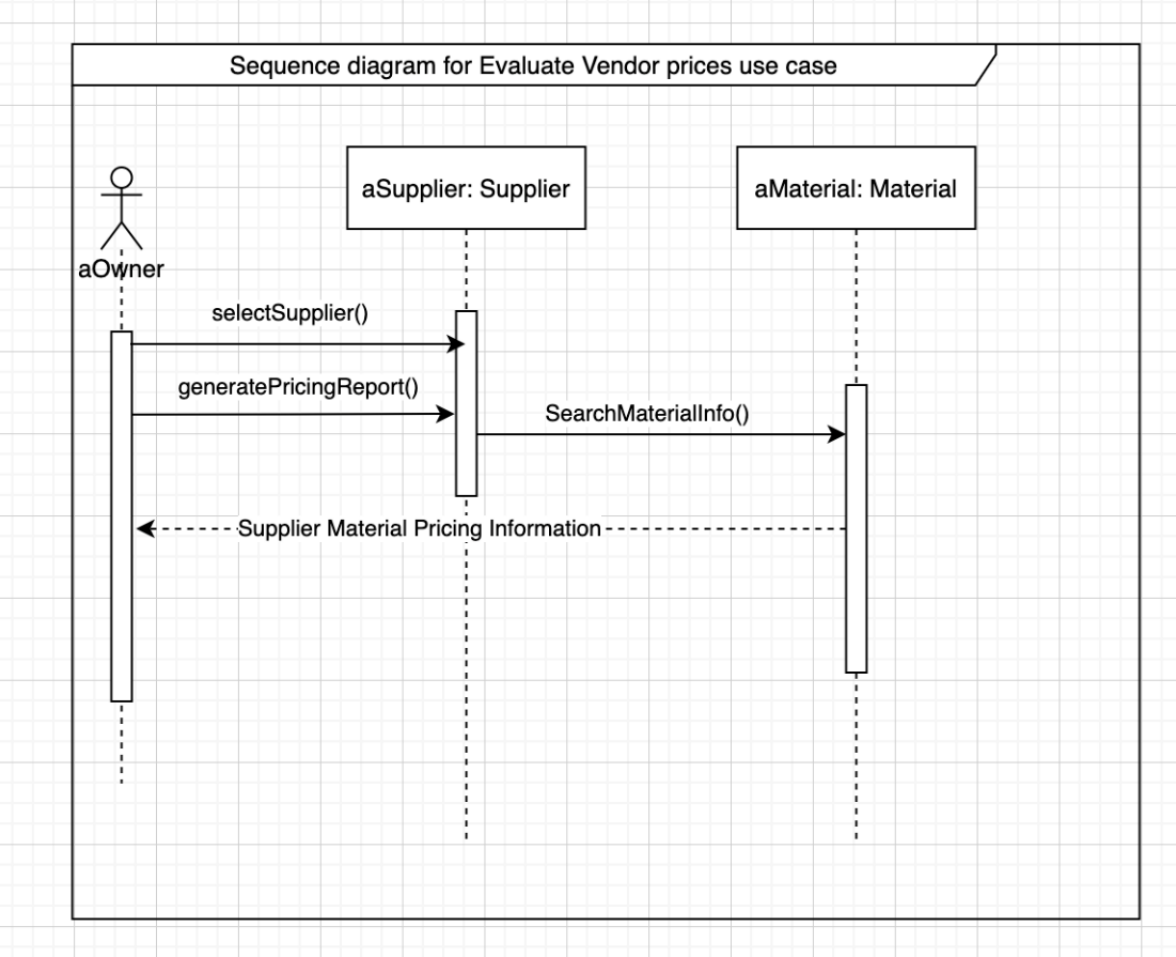
Manage sales Sequence diagram



Some adjustments made to the manage sales use case are:

- BrowseItems() was replaced by searchProducts().
- SelectProduct() flow of information was relocated from EventSale to EventProductList.
- In the previous sequence diagram, we had aProduct:Product and aSale:Sale as objects. This updated version, we replaced Sale for EventSale. Product object was deleted. Also, we added two new objects aOwner:Owner and aInvoice:Invoice.
- By creating these two new objects, we added new flow of information details that were not in the previous version such as createInvoice(), addCustomer(Customer) and addSale(Sale)

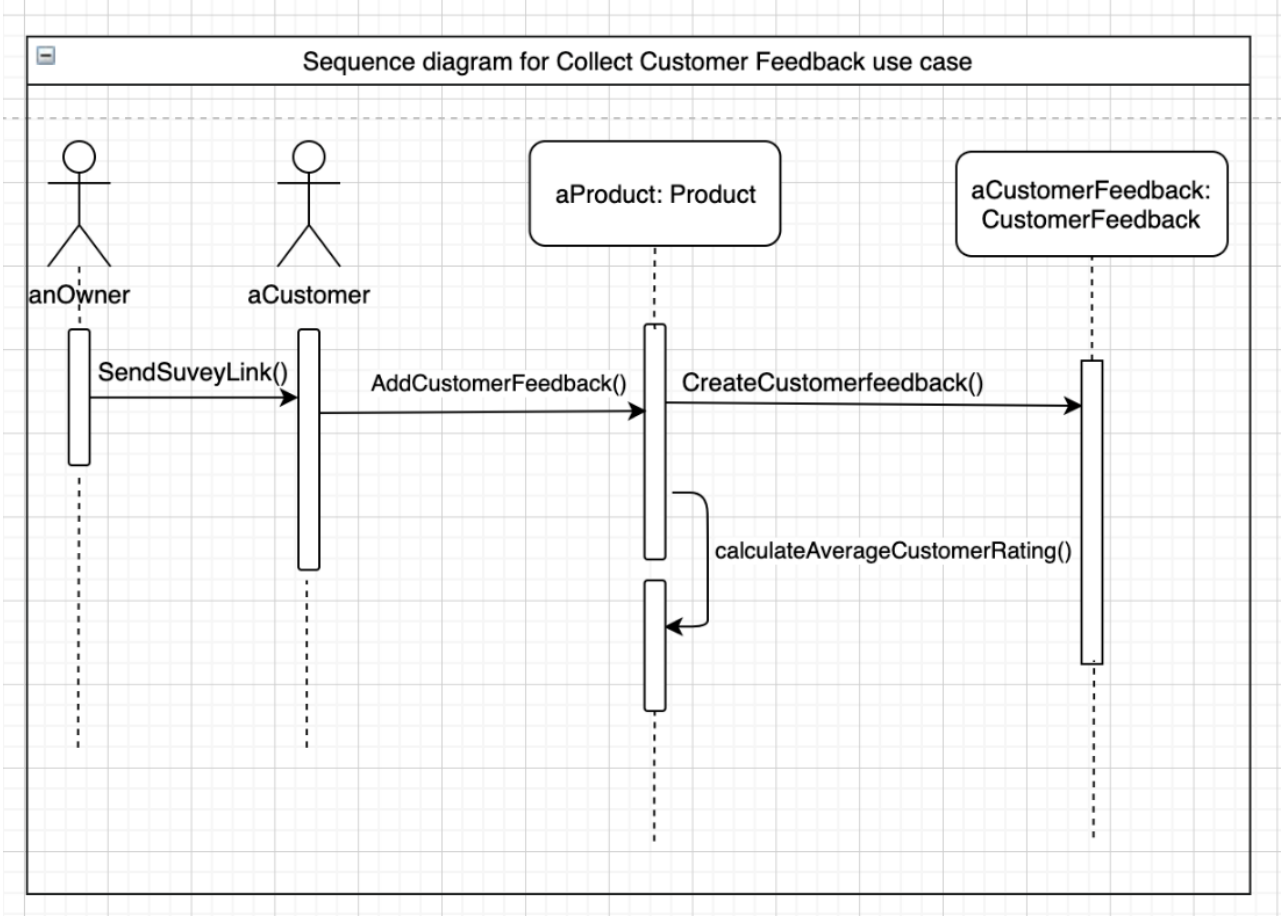
Evaluate vendor prices Sequence diagram



Some adjustments were made to the evaluate vendor prices use case:

- SearchSupplier() has been replaced by SelectSupplier()
- A new message (generatePricingReport()) has been added to the diagram

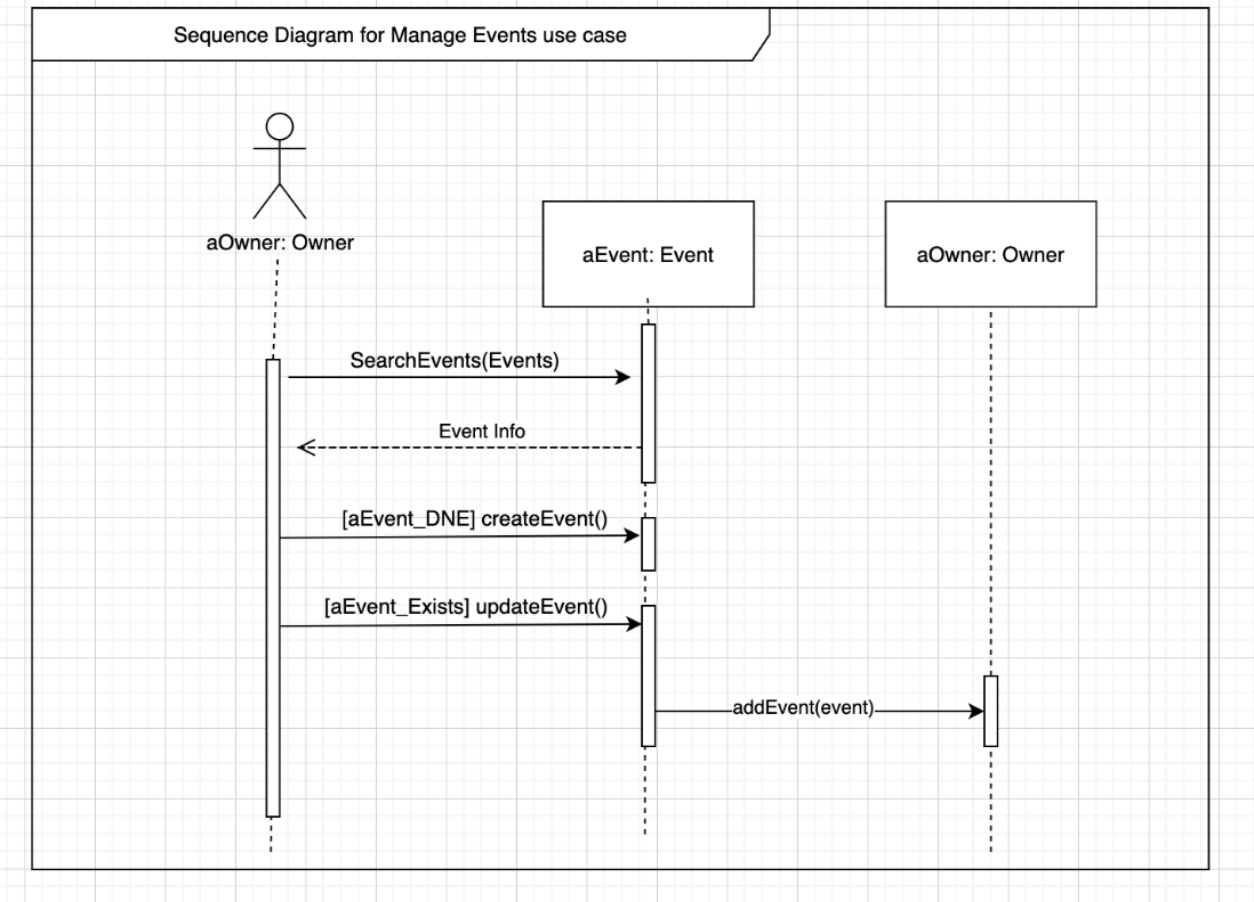
Collect customer feedback Sequence diagram



There were 3 changes made to the collect customer feedback sequence diagram;

- The addition of an owner actor before the customer, and a send survey link information outflow. This was needed because in most instances, the owner is responsible for sending a survey link to the customer to start the feedback process
- We removed browse product details, and all the information outflows and replaced it with addCustomerFeedback(), createCustomerFeedback, and calculateAverage customer rating. This is because our flow for customer details changed after we came up with our Interface diagram. The new sequence had to reflect the customer who receives the link for survey, add his survey, and the system creates and aggregates all the feedbacks.

Manage events Sequence diagram



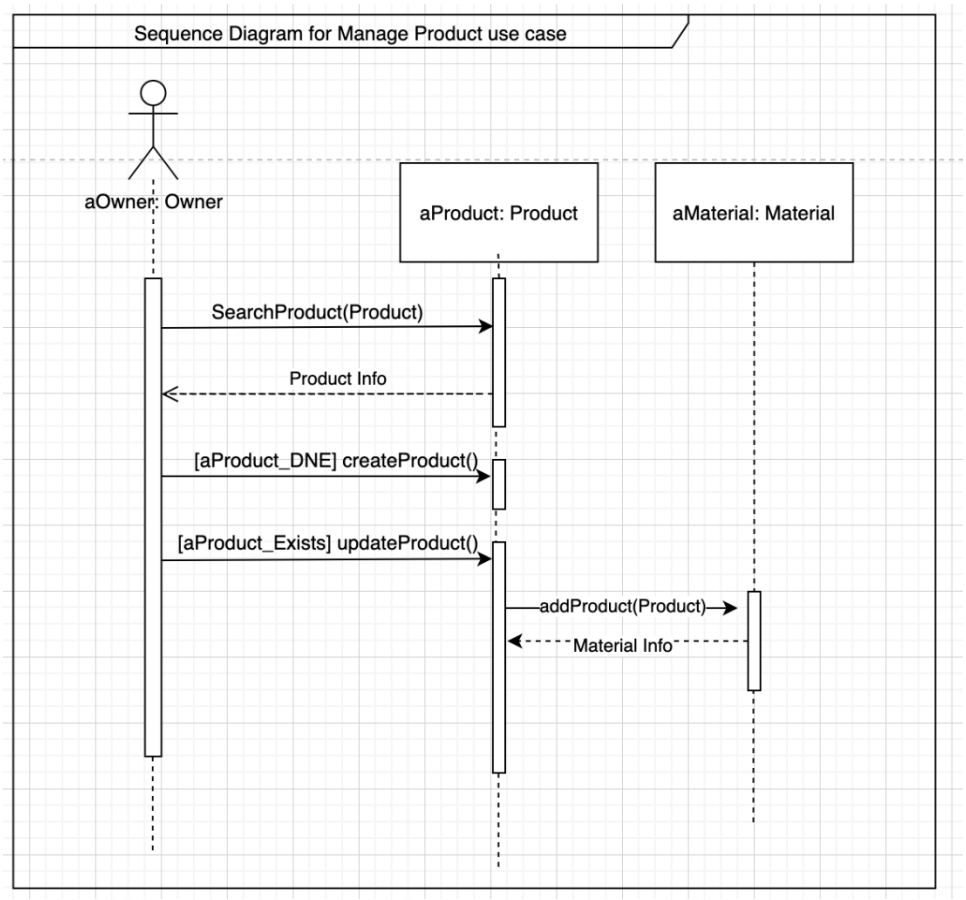
Some adjustments were made to the Events use case;

- The flow of information from the owner to throughout the product selections;

- geteventID() was removed

- A new actor was added aOwner:Owner and an inflow of information was added from Event to Owner addEvent(event).

Manage poducts Sequence diagram



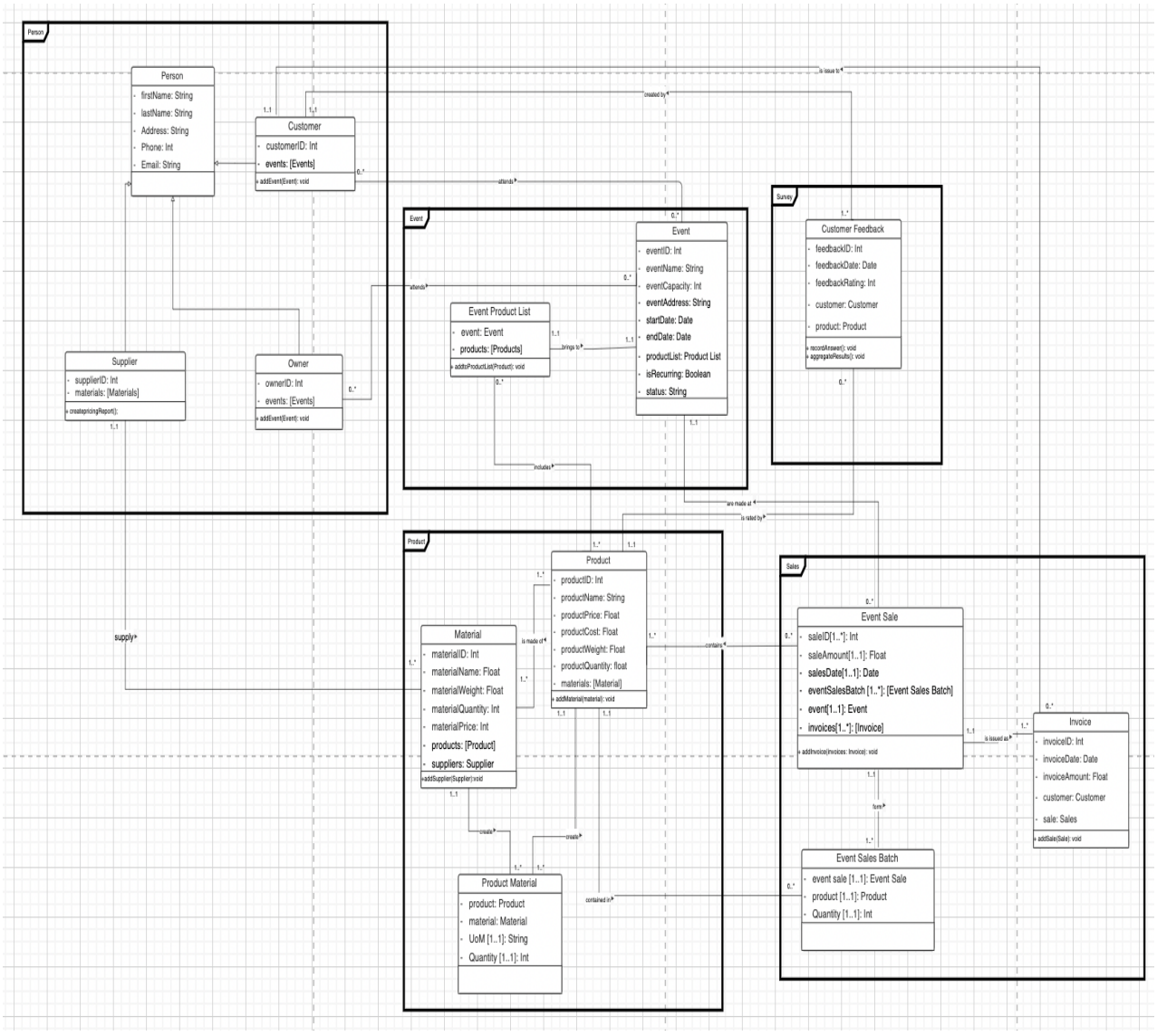
There were 3 changes made to the manage product sequence diagram;

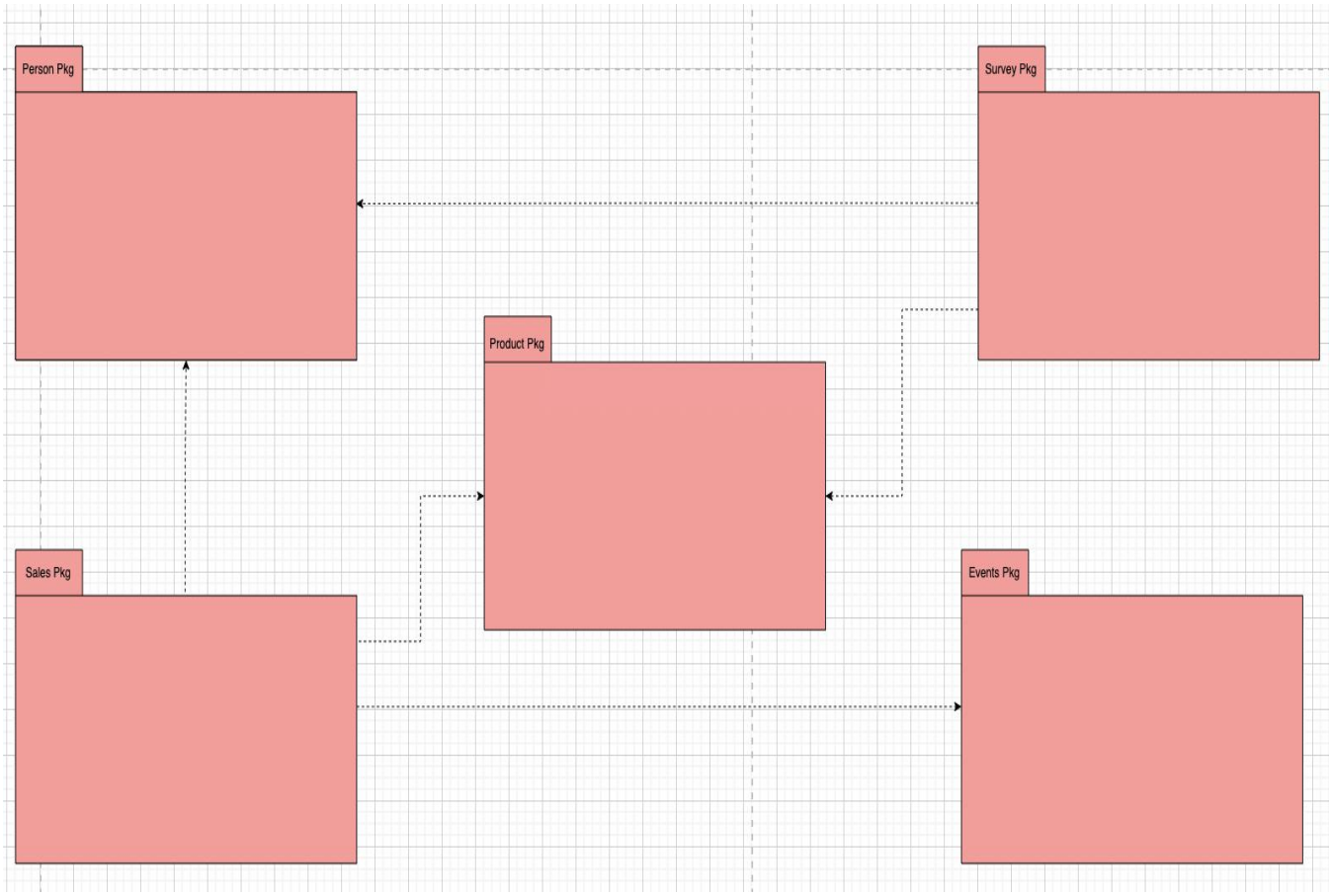
- The actor's name was changed from seamstresses to owner, because after review we figured that to update or add new products, the owner was the one interacting with our system and not the seamstresses. We also didn't have any seamstress class.
- We removed the "get product" information outflow because it didn't need information from the "create product" to update "product" information.
- We modified the "SearchMaterialinfo(product)" line "to add product" because it didn't capture the actual flow information that needed to be done to for the product to be linked to material or fabric

Package Diagrams

A package diagram is a grouping of related UML elements such as diagrams. Documents or classes. It is used to demonstrate the organization and arrangement of various model elements and to reduce complexity of models.

The figures below show our package diagram. It is important to note that no change has been made on this milestone.




Person Pkg
Survey Pkg
Product Pkg
Sales Pkg
Events Pkg

Class and Method Design

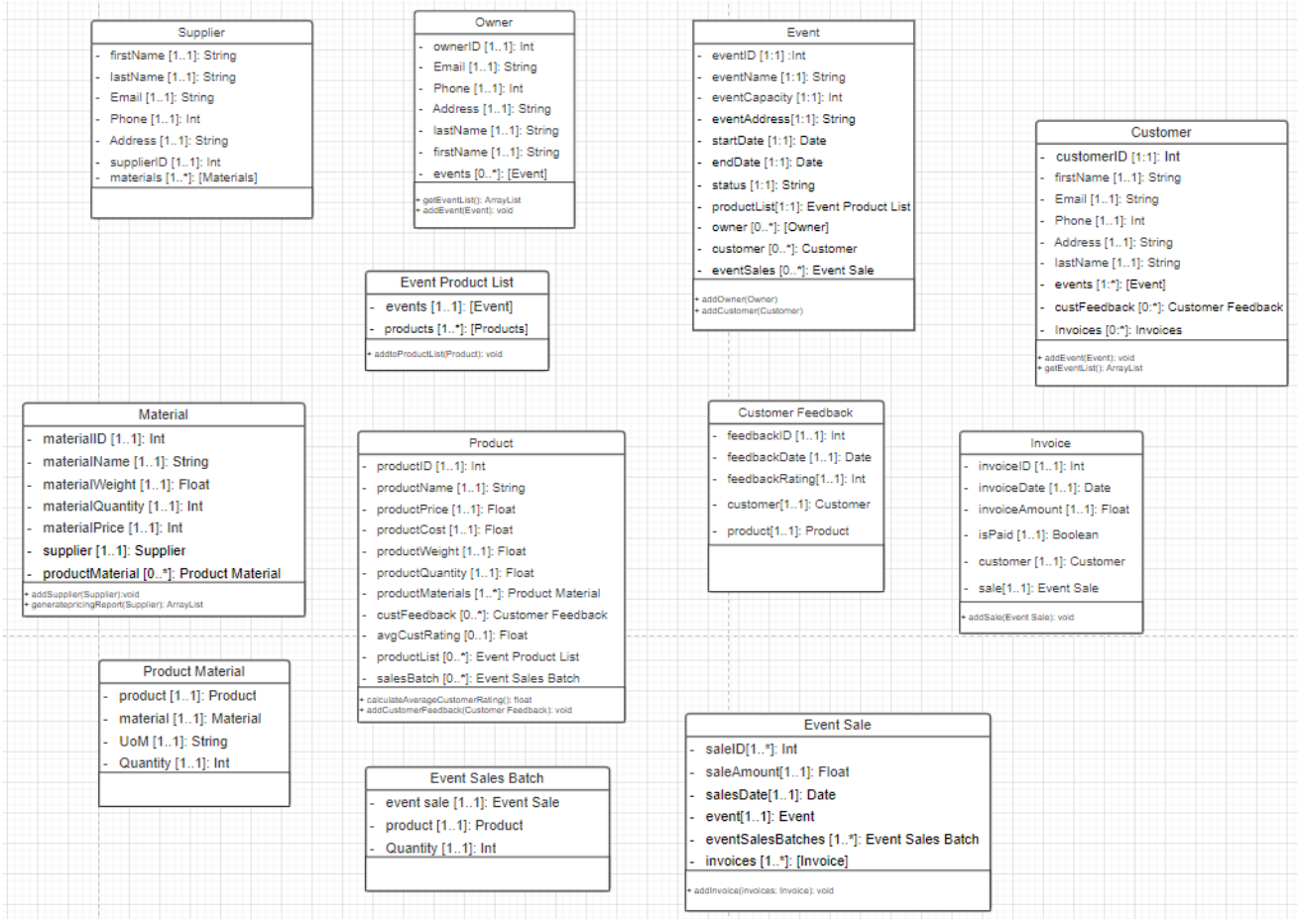
The Class and method design are part of the design phase and are meant to bring have a deeper understanding and clear guidelines to programmers when it comes to describe what the system must do.

Notes on revisions made to the class diagram

There were minor updates that were brought, and these modifications were all brought to the advanced structural model. These changes are;

- The correction from events to event type for the events attribute under Customer class
- The update of customer type from Owner to Customer, under the event class
- The update from sale type to event sales in the addsale method under the Invoice class.
- The addition of customer feedback type to the method addcustomerfeedback() under the product class.

Advanced Structural



Method Contracts

Method Name: addEvent	**Class Name**: Owner	**ID:** 1
Clients (Consumers): Owner		
Associated Use Cases: Manage Events		
Description of Responsibilities: Implement the necessary behavior to add the event object to the list of events the owner and customer will attend / have attended.		
Arguments Received: aEvent:Event		
Type of Value Returned:void		
Pre-Conditions: - The event should exist - The event should not be in the list of events in a specified class instance		
Post-Conditions: - The event object should be in the list of recorded events for the specified instance		

Method Name: addSale	**Class Name: Invoice**	**ID:** 2
Clients (Consumers): Owner		
Associated Use Cases:Manage Sales		
Description of Responsibilities: Implement the necessary behavior to add a sale to a corresponding invoice object.		
Arguments Received: aSale: Sale		
Type of Value Returned: Void		

Pre-Conditions: - A sale must be made in order to add it - Invoice must be created
Post-Conditions: - The sale object must be in the list of corresponding sales for the specified invoice.

Method Name: addCustomerFeedback	**Class Name:** Product	**ID**:3
Clients (Consumers): Customer		
Associated Use Cases: Collect Customer Feedback		
Description of Responsibilities: Implement the necessary behavior to add customer feedback to the list of customer feedback in the product that was rated.		
Arguments Received: aCustomer Feedback: Customer Feedback		
Type of Value Returned: Void		
Pre-Conditions: The customer presses submit on completed feedback form.		
Post-Conditions: The customer feedback object should be in the list of customer feedback for the specific class instance.		

Method Name: generatepricingReport	**Class Name: Supplier**	**ID:** 4
Clients (Consumers): Owner		
Associated Use Cases: Evaluate vendor prices		
Description of Responsibilities: List of suppliers that are strictly associated with their proper material. View available material by supplier basis with prices.		
Arguments Received: aSupplier: Supplier		

Type of Value Returned: Array list
Pre-Conditions: - Supplier must exist - Existing supplier must be associated with a material supplied
Post-Conditions: Return message successful

Method Specifications

Method Name:addSale()	**Class Name:** Invoice	**ID:** 1.0
Contract ID:2	**Programmer:** Chaimaa Nazrati Mariam Henriette Traore	**Date Due:** July 28th, 2021
Programming Language: □ **Visual Basic** □ **Smalltalk** □ **C#** □ **Python**		
Triggers/Events: When an invoice is created following a customer confirming payment		

Arguments Received: **Data Type:**	**Notes:**
Sales Object	The sale object is to be added to the sales list in the invoice object

Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument** **Data Type:**	**Notes:**
Sale.addInvoice()	Invoice	

Argument Returned: **Data Type:**	**Notes:**
Void	

Algorithm Specification:

If an aSale is not already occupied in the invoice as the Sale attribute, Add aSale to Sale Sale.addInvoice() Check if sale is added to the sale list in the invoice instance.
Misc.Notes:

Method Name: addEvent()	**Class Name:** Owner	**ID**: 2.0
Contract ID: 1	**Programmer**: Jonathan	**Date Due:** July 28th, 2021
Programming Language: □ Visual Basic □ Smalltalk□ C# □ Python		
Triggers/Events: When an owner presses a button “attend” when an event is selected.		

Arguments Received: **Data Type:**	**Notes**:
Event Object	Adds the event object to the list of events the owner will go to.

Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument Data Type:**	**Notes:**
Event.addOwner()	Owner	Adds owner to event list of owners

Argument Returned: **Data Type:**	**Notes:**
void	

Algorithm Specification:
If the event exists and the event is not in the list of events in a specified class instance,

Add events to the owner events.
Event.addOwner(aOwner)

Check if the event is added to the event list in the owner class instance.

This method specification sheet applies to the "Customer" class.

```
// Event.addEvent()
Events.add(new Event(eventID, eventName, eventCapacity, eventAddress, startDate,
endDate, status, productList, isRecurring)); //
```

Misc.Notes:

Method Name:generatePricingReport()	**Class Name:** Material	**ID:** 3.0
Contract ID:4	**Programmer:** Estefania	**Date Due:** July 28th, 2021

Programming Language:

□ **Visual Basic** □ **Smalltalk** □ **C#** □ **Python**

Triggers/Events:
When the owner enters a supplier and presses the pricing Report button.

Arguments Received: Data Type:	**Notes:**
Supplier Object	

Messages Sent & Arguments Passed: ClassName.MethodName:	**Argument Data Type:**	**Notes:**
Material.materialPrice()	Material	

Argument Returned: Data Type:	**Notes:**
Array List	

Algorithm Specification: If a supplier exists, and the supplier is associated with a material, Material.materialPrice() for all objects in aSupplier Return all materials with prices in a list for specified supplier. Check if return is successful.
Misc.Notes:

Method Name:addCustomerFeedback()	**Class Name:** Product	**ID:** 4.0
Contract ID:3	**Programmer:** Clive Dency	**Date Due:** 28-07-21

Programming Language:

□ **Visual Basic** □ **Smalltalk** □ **C#** □ **Python**

Triggers/Events:
A customer submits complete customer feedback on a product

Arguments Received: **Data Type:**	**Notes:**
Customer Feedback Object	

Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Argument** **Data Type:**	**Notes:**
CustomerFeedback.new()	Customer Feedback	
calculateAverageProductRating()	Product	

Argument Returned: **Data Type:**	**Notes:**
Void	

Algorithm Specification:
When a customer clicks submit on a feedback form,
CustomerFeedback.new(feedbackID, feedbackDate, feedbackRating, aCustomer, aProduct)
Product.calculateAverageCustomerRating(aProduct)
Add customer feedback to the customer feedback list in the product.
Check if the customer feedback has been added to the customer feedback list in the product.

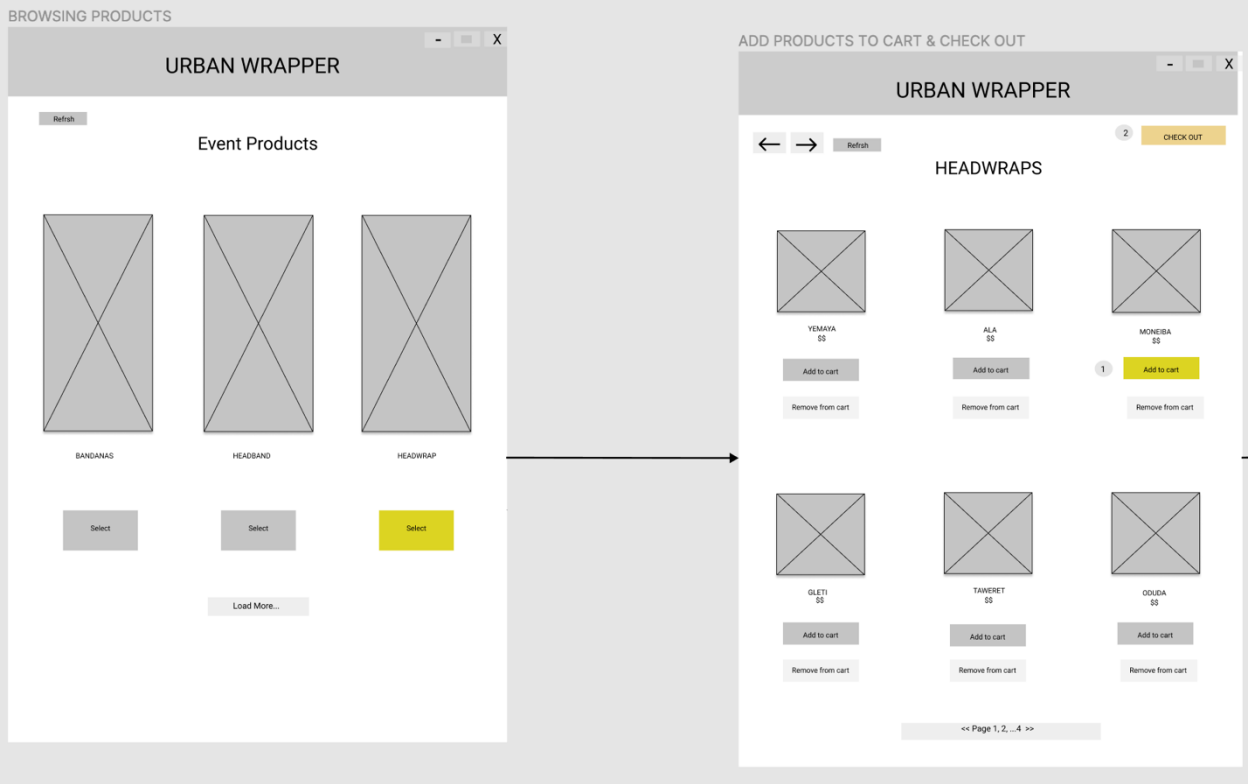
Misc.Notes:

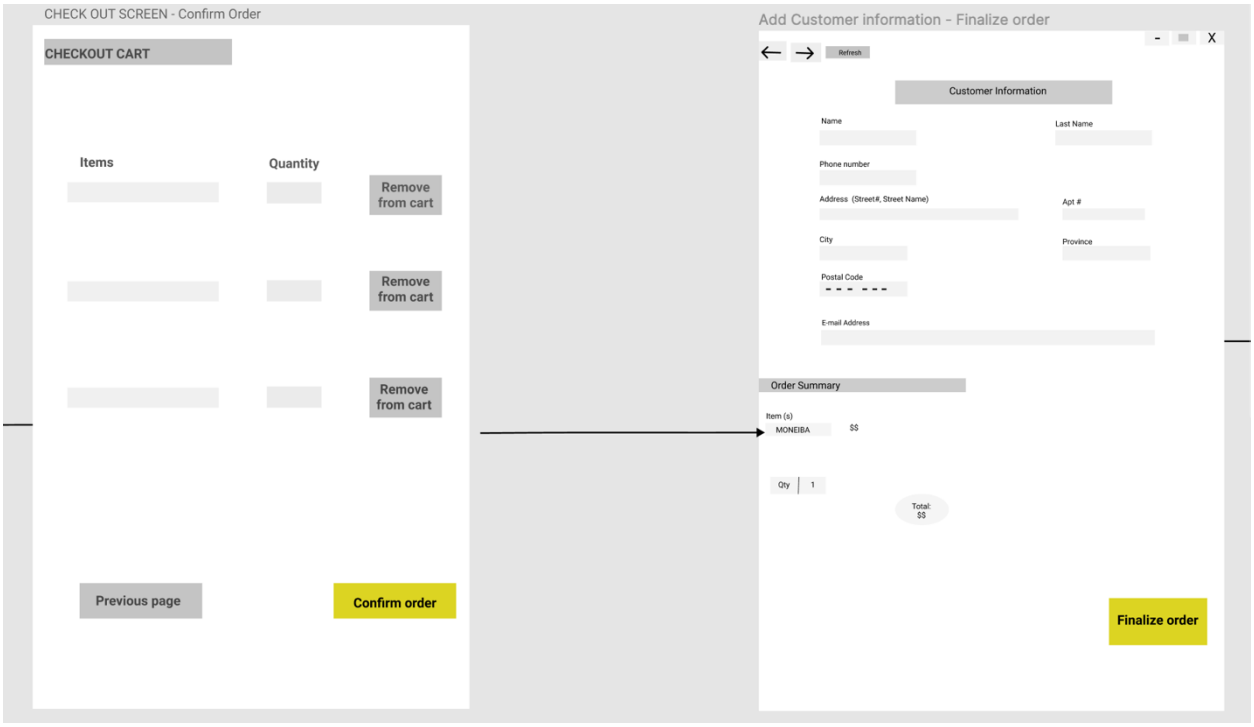
HCI Design

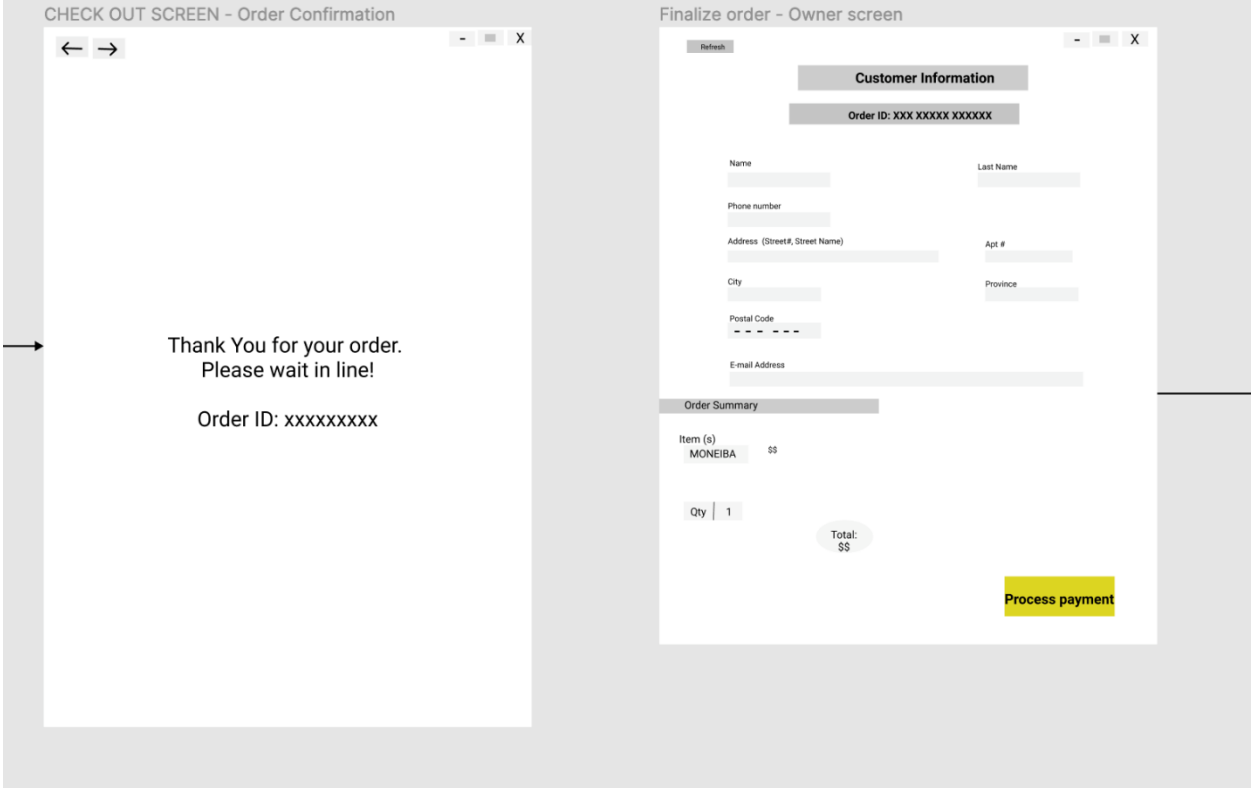
The Human Computer Interaction (HCI) studies how people interact with technology interfaces such as computers, cellphones, or any other device. In our case, the user interacts with an iPad. Our goal was to design an interface for each use case that will be simple, accessible, and useful to users.

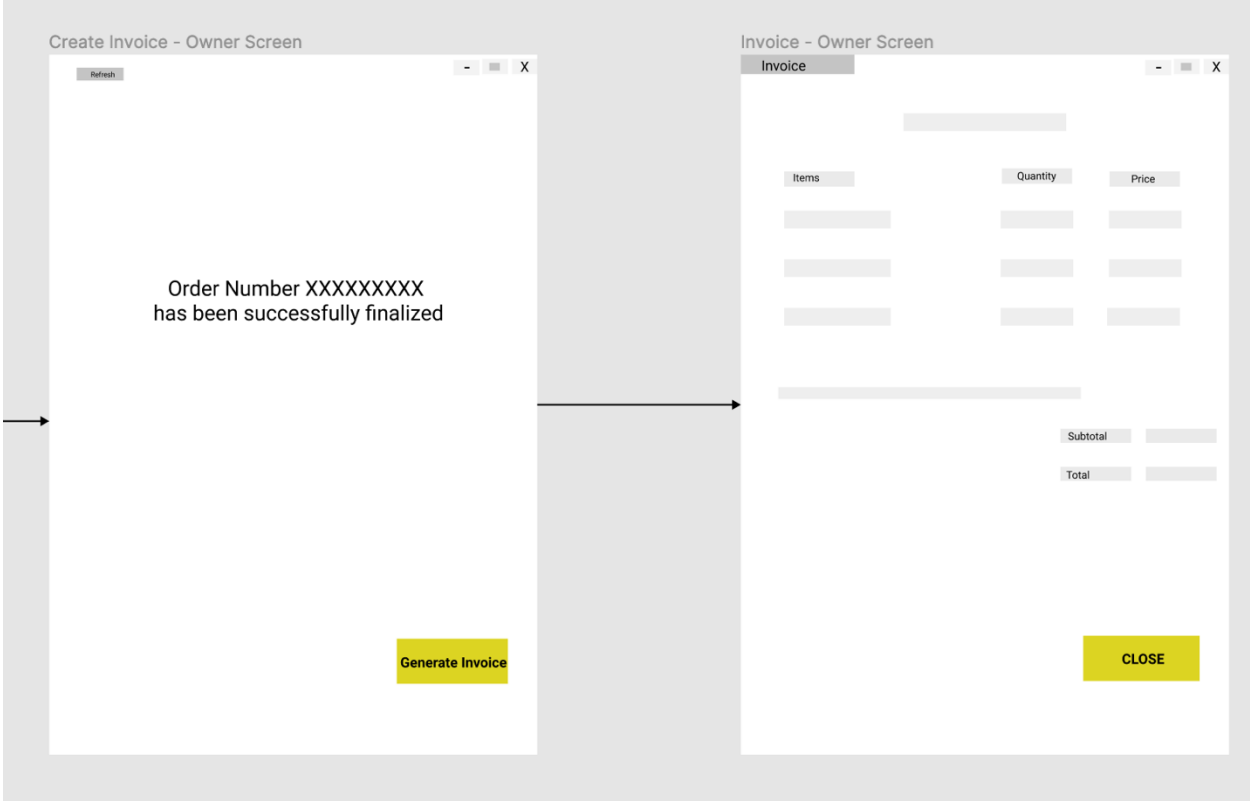
As shown in the figures below, the HCI Designs created for this project are based on the 3 chosen use cases, under each use case there are multiple wireframes that interacts with each other through Wireflows. These wireframes showcase how the interface would look for each use case.

Manage Sales




CHECK OUT SCREEN - Confirm Order
CHECKOUT CART
Items
Quantity
Remove from cart
Remove from cart
Remove from cart
Previous page
Confirm order
Add Customer information - Finalize order
Refresh
Customer Information
Name
Last Name
Phone number
Address (Street#, Street Name)
Apt #
City
Province
Postal Code
E-mail Address
Order Summary
Item (s)
MONEIBA
$$
Qty
1
Total:
$$
Finalize order


CHECK OUT SCREEN - Order Confirmation
Thank You for your order.
Please wait in line!
Order ID: xxxxxxxxx
Finalize order - Owner screen
Refresh
Customer Information
Order ID: XXX XXXXX XXXXXX
Name
Last Name
Phone number
Address (Street#, Street Name)
Apt #
City
Province
Postal Code
E-mail Address
Order Summary
Item (s)
MONEIBA
$$
Qty
1
Total:
$$
Process payment



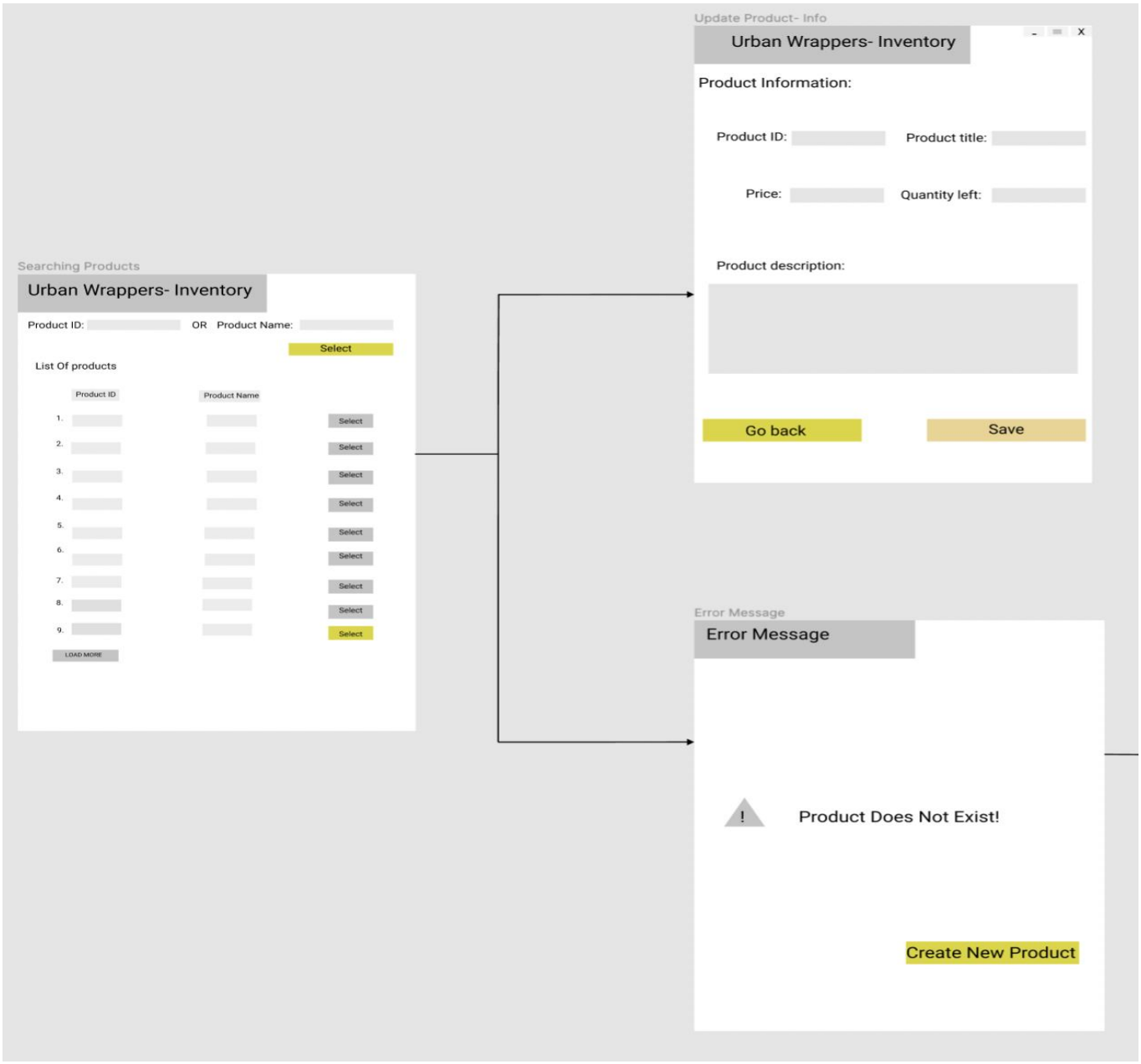
User story – Customer Interface

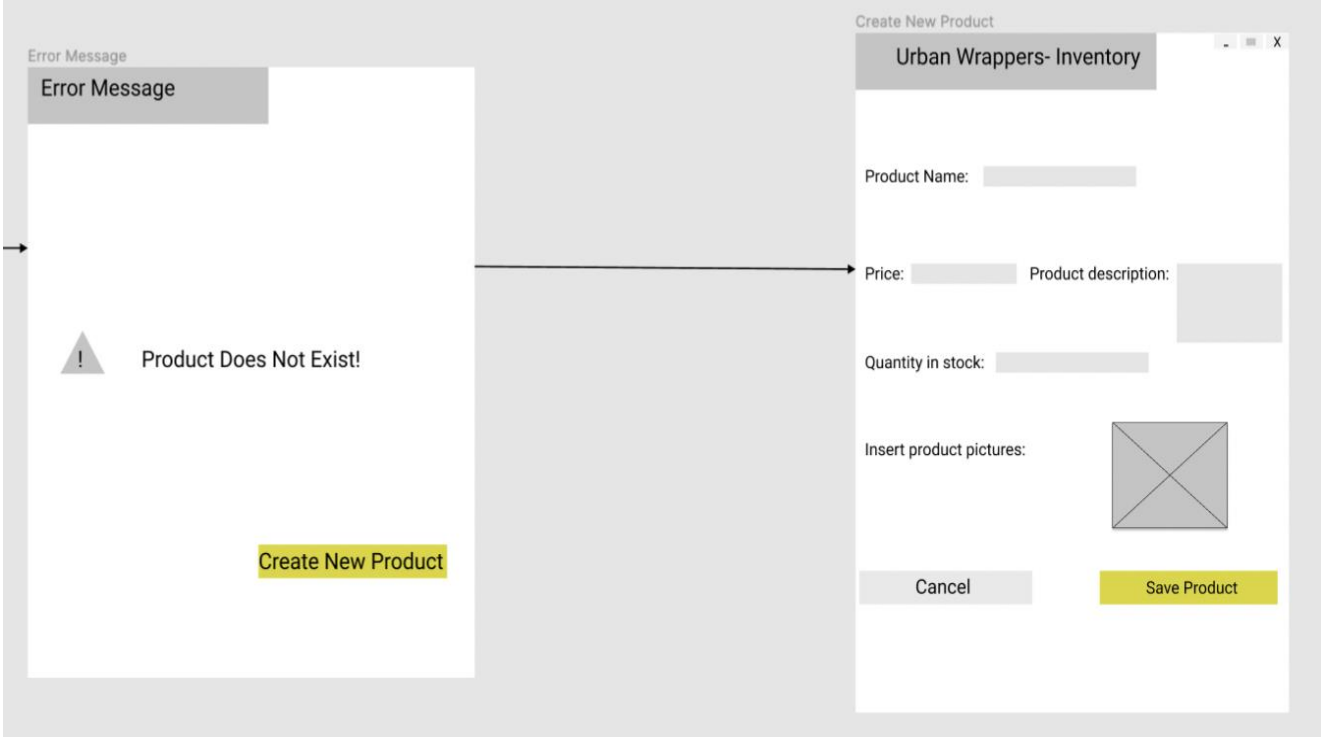
As a customer
I want to be able to browse products
So that I can order products

Scenario – Customer Browses Products

Given I'm a customer
When I open the event products page
Then the system displays the different products that will be available at the event
When I click on "load more"
Then more products are displayed
And I click the "select" button
And I click the "Add to cart" button or "remove from cart" button
Then the system shows me a review of my cart with the option to remove the product from the cart
And I click the "confirm order" button
Then the system shows the customer information page and the order summary
And I enter my information
And I click the "Finalize order" button
And the system shows a message saying "Thank you for your order.
Please wait in line! Order ID: xxxxxxxxx"

Manage Products





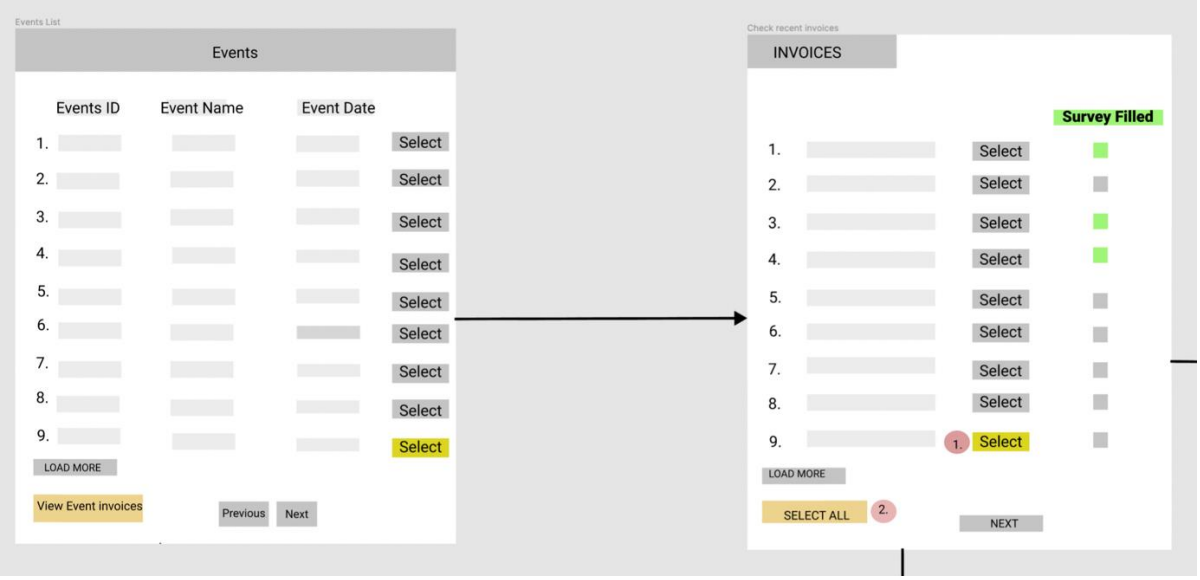
Owner Interface:

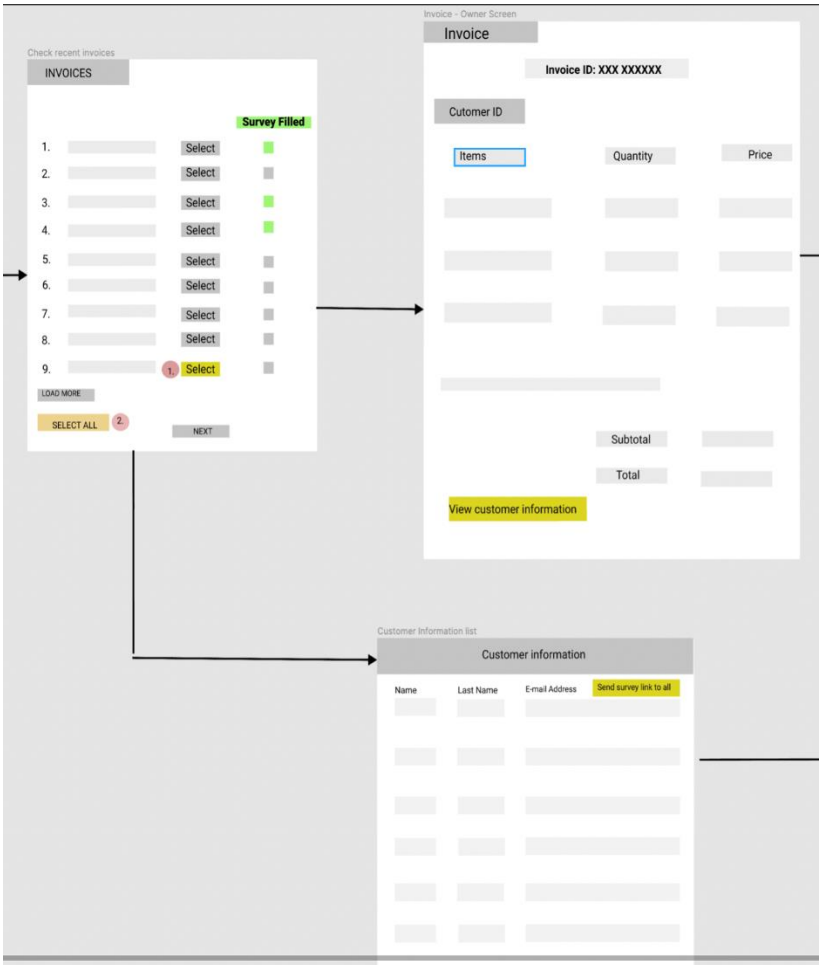
As an owner
I want to be able to add or update the information of my products
So that I can add new products in my inventory or update current product information

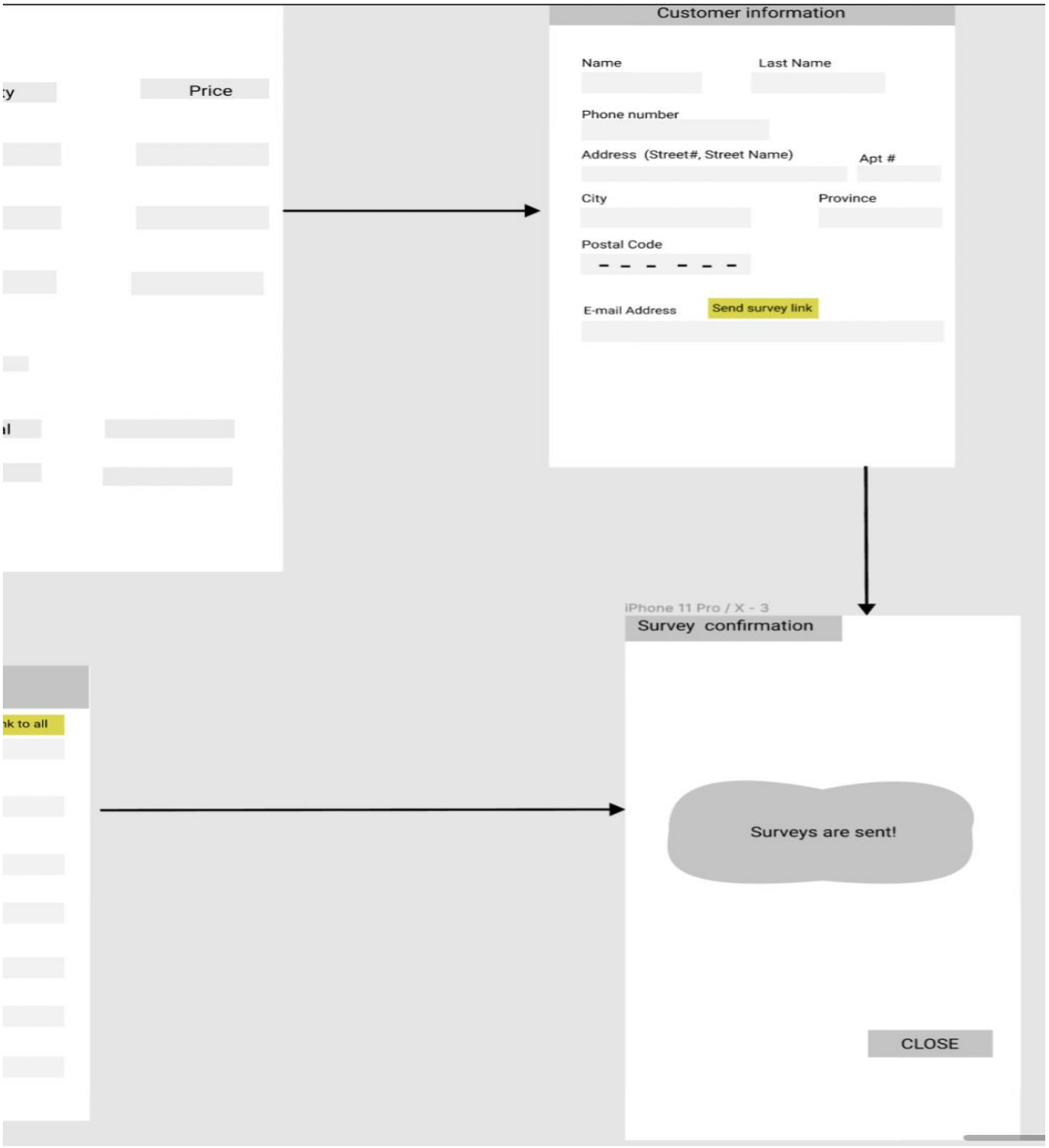
Scenario:

Given that I have the role of an owner,
When I type in the Product ID and product Name or when I select a product from the product list
If the product exists, the system displays the product information page
When I change the information and click on "save"
Then the product information is updated and saved.
If the product does not exist,
The system displays an error message saying "The product does not exist"
I then click on "Create a new product"
And the system will lead to the "create new product page"
I then fill in the product name, price, quantity, description and image
And I click on "save product"
Then the system will save the product.

Collect Customer Feedback







Customer interface user story:

As a customer
I want to be able to fill a survey
So that I can give feedback on the products

Scenario:

Given I am in a role of a past customer
When I click on "Take survey"
Then the system shows a list of multiple-choice questions
And the system shows a comment section at the end

When I fill in the comment section and I click on “Finish”
The system shows a page with a text saying “Thank you for your time and we hope to see you again soon!”

Owner interface:
As a website user
I want to be able to send a survey link to customers
So that I can receive feedback on my products

Scenario:
Given I am in a role of an owner
When I want to send survey links of a specific event
I will need to click on “select” followed by “View event invoices”
Then the system shows the list of invoices

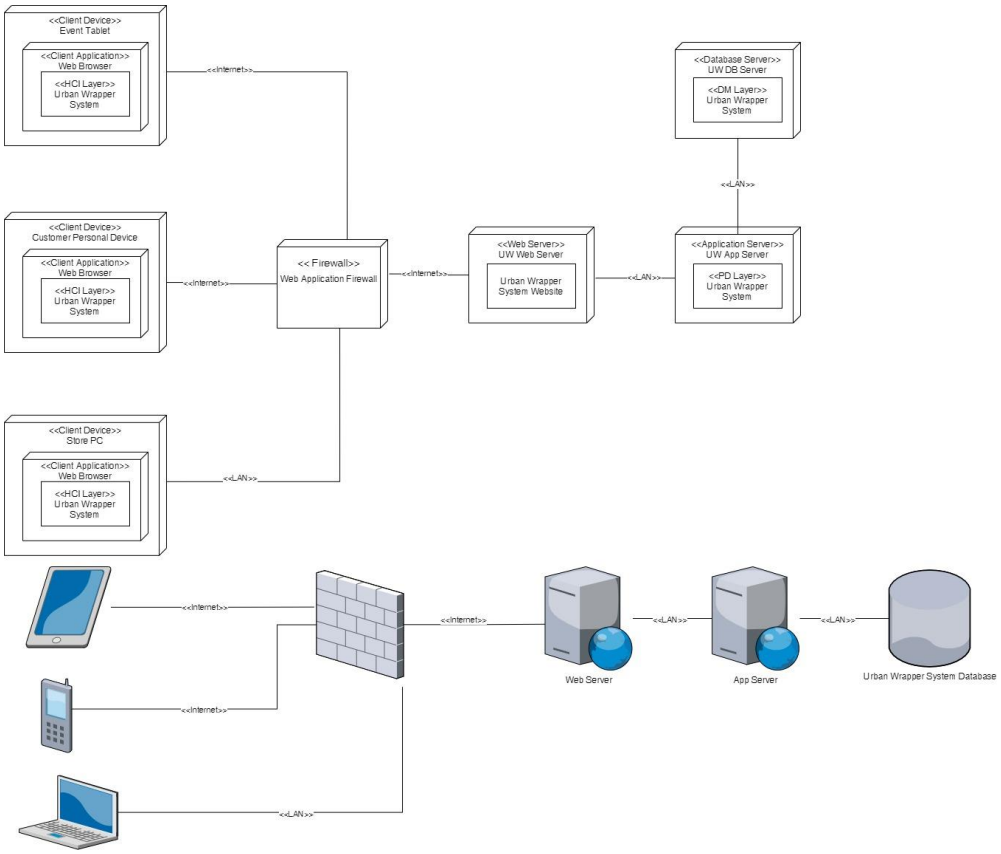
Option 1:
When I want to get a single customer in formation, I click on “Select”
And the system shows me a customer information
Then I click on “send survey link”
And the system then shows a page with a text saying “ survey is sent!”

Option 2:
When I click on “Select all”
The system shows a list of all customer information
I then click on “send survey link to all”
And the system shows a page with a text saying “survey is sent!”

Deployment Diagram

The diagram below shows the architecture of our system.

- Our application supports three types of clients: an event tablet, customer's personal device and a local desktop pc in the store.
- There is a web application firewall which protects and secure web applications from unauthorized access and attack attempt.
- Next, there is a node web server which host the front end of the application providing the user interface.
- Then, the application server node which is for the back-end component. Thus, it implements the business logic. For instance, in our application, this is about managing the product catalogue. Managing sales and managing customer feedback.
- Internet available is needed for client devices (an event tablet, customer's personal device) to access the web server. In addition, a Local Area Network (LAN) to have access from server to server as well as LAN for the local desktop pc to access the web server.
- Finally, the database server manages the persistence of the data model. Meaning that, the data cannot be deleted by external processes or objects until the user deletes it.



Prototype

Github Link

https://github.com/Groot99/UrbanWrapper

Data Management

We used a local MySQL server for data management. Database schema in an SQL script was created, generated, and imported to Microsoft Visual Studio using SQL Server Management Studio, mapping tables and attributes accordingly and appropriately for the prototype.

Physical Architecture

The prototype and an SQL localhost to pair were created using consumer-level Windows-based computers. Refer to the hardware and software specifications / deployment recommendations below for the final launch.

Standard Client – Tablet Events

Operating System

Android / iOS
(Web browser)

Special Software

Web browser – Google Chrome Recommended

Hardware

- 4GB Memory
- 256GB Storage
- Tablet / iPad

Network

Mobile Data 50MBpS

This is a basic tablet that will be used to access the application from events. Equipped with mobile data to access the application over the internet and enough hardware to efficiently run the web application without delays.

Standard Client – Owner / Administrator

Operating System

- Windows
- (Web browser)

Special Software

- Web browser – Google Chrome Recommended

Hardware

- 8GB Memory
- 512GB Hard Drive Storage
- Intel Core i5
- 21” Monitor
- Mouse / Keyboard

Network

- Ethernet 75MBpS

The device inside the business that the owner will use to regularly interact with the application. Contains hardware to accommodate for the owner’s business-related files and activities. The device will be connected to the server via ethernet.

Standard Web Client: *Hosts the website*

Operating System

- Linux
- Special Software
- Apache

Hardware

- 16GB Memory
- 512GB Hard Drive Storage
- Intel Xeon E5-2670
- 21” Monitor
- Mouse / Keyboard

Network

- Ethernet 75MBpS

Linux based with Apache to host the application to those outside of the shop – for owners at events, and customers from home. With hardware to accommodate for a growing userbase, upgrades should not be required for some time. 8 core processor at low cost.

Standard Application Server: *Hosts the application*

Operating System

- Linux

Special Software

- ASP.NET Core Framework

Hardware

- 16GB Memory
- 512GB Hard Drive Storage
- Intel Xeon E5650
- 21” Monitor
- Mouse / Keyboard

Network

- Ethernet 75MBpS
- Linux based with .NET Core Framework to run the C# application cross-platform, over a web browser. Enough hardware to support a growing number of users, and an affordable CPU.

Standard Database Server: *Hosts the database server*

Operating System

- Linux

Special Software

- MySQL
- ASO.NET

Hardware

- 16GB Memory
- 2 1TB Hard Drive Storages
- Intel Xeon E5650
- 21” Monitor
- Mouse / Keyboard

Network

- Ethernet 75MBpS

Linux based with MySQL to support the database of the system. Abundance of RAM to ensure fast processing well into the future as the database grows. Two large hard drives to maintain regular backups and easy replacement in case of failure. Affordable CPU.

A standard client that should also be considered is the customer from their personal devices. - As the application is intended to be cross-platform. Accessible from a browser with low operating requirements, there are little to no additional hardware specifications.

Our deployment decisions not only accommodate for future growth of the business, but also provide stability for present day-to-day operations through appropriately fitting, cost-effective hardware with enough processing power to efficiently operate presently and well into the future. The hardware specified here provides support for the launching of our application.

Construction

- C#

The language in which the prototype is coded in. The most familiar programming language for the programming team, therefore easiest to work with.

- Visual Studio Community 2019

Integrated Development Environment. Visual Studio Community was chosen for the programming team's familiarity, preference, and proficiency.

- .NET Framework

The .NET framework is required to create Windows applications. This was the optimal framework to create a functional prototype in this case. The final product will be using ASP.NET framework, enabling the application to work via web browser.

- ADO.NET Entity Framework

A library that provides ease of access between databases and .NET framework applications programmed in C#.

- Windows Forms

Provides GUI capability for the prototype in C# for Windows operating systems.

The programming team perpetually performed an array of system tests throughout the development to ensure the prototype meets functional requirements. Functional and non-functional requirements of the final system are as follows:

Operational Requirements

- The system will work over the web environment with Google Chrome.
- All office locations will have an always-on network connection to enable real-time database updates – owner will be able to monitor sales in real time.
- A version of the system will be provided for desktop, tablet, and mobile users.
- The system must be able to work with various operating systems – Linux, OSX, Windows.
- The system must be optimized to work with minimal bandwidth – over limited mobile data if required (owners attending events - may be remote)
- The system must be able to operate with handheld devices and tablets – owner, customers use tablets and smartphones.
- The system will be able to support additional locations with six months' advance notice.
- The system will be able to integrate to systems and databases created with six month's advance notice.

Performance Requirements

- Response time must be less than 7 seconds for any page / database submission over the network.
- There will be a maximum of 250 simultaneous users at peak use times.
- A typical transaction of data / page submission will require approximately 10kb of data bandwidth.
- Scheduled maintenance shall not exceed a day.
- The system shall have 99% uptime performance.

Security Requirements

- The system is not mission critical, but an outage is estimated to cost $1,500 per hour on average during an event.
- A complete loss of all system data is estimated to cost $25,000.
- Those not logged into the system may only access customer feedback page instances.
- Only owners can read and modify the system and database entries.
- Users are required to log into the system to authenticate.
- Personal information must be secured and encrypted.

Cultural & Political Requirements

- Personal information about customers cannot be shared or transferred.

Installation and Operation

To install and operate the working prototype, the prototype must be configured to an accessible SQL server through the application configuration. The SQL server must also be populated with tables required for operation. These can be created by executing the included SQL script.

Upon the owner's final confirmation of the finished system and its conformance to the business standards and culture, the programming team will implement the system and required software on the store's hardware as specified. As there are no systems in place, it will be a direct, simultaneous, whole-system conversion approach. This approach leverages the implementation's low-risk and resistance to change due to the scale of the business and committed stakeholders that are motivated to adopt. The programming team will be responsible for maintenance of the system for the next two years.

User Testing and Training

- The prototype was perpetually tested throughout the development process.
- The system will be thoroughly tested on the store network and via the internet following final implementation of the system. The owner may participate in troubleshooting if required to build skills.
- The finished system UI will be configured for ease of access to customers. No training for customers required.
- The owner will be given a full-day training session following the implementation of the system. – Additional support will be provided within a day of notice.

Appendix

Appendix 1: Software Used

In order to manage our team, draw our models, and build our application, a variety of Software programs. The following were the software used to support our work;

- Draw.io:

- Figma:

- Lucid Chart:

- Visual studio:

- SQL server management studio

- Jira Project Tracking

BTM 495- Team Contract

1. Meeting Coordinates
    1. Location: Teams, Zoom or Messenger Meeting
    2. Day: Monday, Tuesday & Wednesday (but not excluding the other days of the week)
    3. Time: 6:00 PM EST
    4. Duration: 3-6 hours
    5. Additionally on Friday, we will meet at least once a week with the professor
    6. If any other meetings are required within the scope of the project, the team will find an appropriate time slot during the following weekend.

2. Responsibilities of each team member

    7. Responsibilities and rights:
        - o Managing the discussions
        - o Managing time and schedule
        - o Managing project tasks
        - o Approval of Meeting minutes document for current week
        - o Preparing the agenda for the meeting
        - o Taking notes during the meeting.
        - o Complete Meeting minutes document
        - o Organizing and sending the notes from a meeting in a minute's template
        - o Consequences of failure to carry out responsibilities will see him deducted 10% off of his final peer evaluation assessment.

2.1. Members

1.Rights
- o Equal treatment
- o Majority vote
- o Provide feedback, express opinion, and get help from team members

2.Responsibilities
- o Respect team members and majority's decision
- o Being punctual
- o Follow contract rules
- o Complete tasks on time
- o Consequences of failure to carry out responsibilities will see him deducted 10% off of his final peer evaluation assessment.

3. Performance Expectations at Team Meetings

3.1Attendance and Punctuality

8. Attendance will be defined as follows: presence during online meetings (Zoom). Team members must be present at 80% or more of all meetings.
9. Valid reasons for absences: class conflict, technical connectivity, religious purposes.
10. Time/day for advance notice of absence before the meeting date: the absentee will notify the rest of the team a minimum of 2 hours in advance.
11. Maximum number of absences allowed with prior notification: the absentee will be allowed a maximum of 2 absences without prior notification.
12. Consequences for unacceptable/unexcused absences: 10% deduction off peer evaluation
13. An absence, whether it is excused or not, should not be a reason not to perform the task the team member has agreed to take on. The absentee should do his best to inform the team of his work despite his absence during the meeting.
14. Define lateness: a member will be considered as late if he arrives at a group meeting 20 or more minutes late.
15. Consequences for repeated lateness: 10% deduction off peer evaluation (lack of commitment)

3.2. Participation in Discussions and Task Handling:

1. Well-managed collaboration
2. Sufficient preparation
3. Meeting agreed upon deadlines
4. Asking other members for help and providing help when asked
5. Consequences of failure in taking responsibility of project tasks and/or completing them
6. Basis of assessment for the member's work quality

1. Consequences of failure to carry out responsibilities will see him deducted 10% off his/her final peer evaluation assessment.

4. Communication Media:

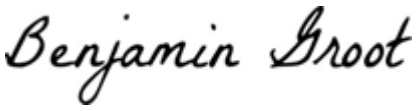
16. Team calls will be done via Teams, Zoom or Messenger.
17. All email related to project work will be forwarded to the entire team for transparency purposes.
18. All project-related files will be stored in a shared Google Drive.
19. Files will be available for modifications by all team members.

5. Guidelines to Improve Meeting Effectiveness:

20. Giving structured feedback to team members during and the end of the project
21. Demonstrating flexibility, responsibility, and motivation

22. Managing differences of opinion and conflicts among members
23. Consequences of domineering behaviour and non-constructive criticisms

Date of Signature: 07/07/2021

Name	Signature
Faisal Alshowaier	
Estefania Carolina Castillo	
Benjamin Groot	
Clive Nya	
Chaimaa Nazrati	
Jonathan Tartaglia	
Mariam Henriette Traore	

APPENDIX 3: TEAM MANAGEMENT

I. MEETING MINUTES:

Date: July 6, 2021 **Meeting Time:** 4:40 PM – 8 PM **Attendees**: All
Work description: • Introduced to each other more formally, and worked on the project proposal due on the 7th of July

Date: July 12, 2021 **Meeting Time:** 4:40 PM – 8 PM **Attendees**: All
• Work on Use case diagrams and activity diagrams

Date: July 13, 2021 **Meeting Time:** 4:40 PM – 8 PM **Attendees**: All
Work description: • Worked on the Structural modeling, discussing about the classes that needed to be included.

Date: July 14, 2021 **Meeting Time:** 4:40 PM – 8 PM **Attendees**: All
Work description: • Worked on completing the Structural modeling, • Completed Class Diagrams and CRC cards

Date: July 19, 2021 **Meeting Time:** 6 PM – 8 PM **Attendees:** All

Work description:

- Worked on design modeling workshop and preparing package diagrams
- Made some adjustments to the Behavioral Modeling

Date: July 20, 2021
Meeting Time: 7PM – 11 PM
Attendees: All

Work description:

- Worked on implementing the changes that were made by the teacher about our use cases
- Worked on the sequence diagram based on the updated structural model

Date: July 21, 2021
6 PM – 8 PM
Attendees: All

Work description:

- Worked on editing Behavioral modeling
- Commenced discussion on UML diagrams and started the packaged diagrams
- Discussed next steps for meeting on July 25th.

Date: July 25, 2021
Meeting Time: 6 PM – 8 PM
Attendees: All

Work description:

- Worked on corrections for structural and behavioural modeling.
- Completed package diagrams.
- Discussed next steps for meeting on Monday 26th.

Date: July 26, 2021
Meeting Time: 4:40 PM – 8 PM
Attendees: All

Work description:

- Worked on corrections for behavioural and structural modeling. Meeting with teacher x2.
- Completed advanced class diagram.
- Revised package diagrams.
- Began working on methods design.
- Discussed next steps for meeting on Tuesday 27th.

Date: July 27, 2021
Meeting Time: 6:00PM – 8:00PM
Attendees: All

Work description:

- Worked on class and methods design milestone (Method specification and contract)
- Revised the advanced Structural Modeling

Date: July 28, 2021
Meeting Time: 6:00PM – 8:00PM
Attendees: All

Work description:

- Revised Use case diagrams with any changes needed to be done
- Modified the structural models
- Modified methods as additions to the application were needed

Date: August 2nd, 2021
Meeting Time: 6:00PM – 7:00PM
Attendees: All

Work description:

- Organized the next steps in the projects
- Looked over all the work until today, made sure everything was up to par
- Divided the group into two teams for the HCI, architecture design and the demo

Date: August 2nd, 2021
Meeting Time: 6:00PM – 7:00PM
Attendees: All

Work description:

- Organized the next steps in the projects
- Looked over all the work until today, made sure everything was up to par
- Divided the group into two teams for the HCI, architecture design and the demo

Date: August 3rd, 2021
Meeting Time: 6:00PM – 9:00PM
Attendees: All
Work description:

- Both Groups worked with their proper team to finish the task at hand
- HCI and Deployment Diagram started to be put together
- Discussion for what language to be used for the code was in the talks (pros and cons) decided to go with Phyton

Date: August 5th, 2021
Meeting Time: 6:00PM – 9:00PM
Attendees: All
Work description:

- HCI and Deployment Diagram started to be put together
- The programing team ran into a roadblock with their choice of language and agreed to switch the language to C# as most members had some experience

Date: August 8th, 2021
Meeting Time: 6:00PM – 9:00PM
Attendees: All

Work description:

- HCI and Deployment Diagram was at the final stages and modification were being made according to the team opinion

- The programing team commenced to develop the diagram, a lot of references made with YouTube and StackOverflow.
- Issues were starting to develop and needed to be delt with before moving forward.
- A lot of programming errors needed to be addressed (very time consuming)

Date: August 9th, 2021
Meeting Time: 2:00PM – 9:00PM
Attendees: All

Work description:

- HCI and Deployment Diagram was at the final stages and modification were being made according to the team opinion (revised one more time before submitting after class
- Addressing all the programming errors that were being apparent from the day before.

Date: August 10th, 2021
Meeting Time: 6:00PM – 9:00PM
Attendees: All

Work description:

- Found out we needed a database to the software being built.
- Discussions on what tool we could use, we had the idea to use Excel but didn't seem to be efficient and moved towards SQL server management after some research being done on YouTube.

Date: August 11th, 2021
Meeting Time: 2:00PM – 9:00PM
Attendees: ALL

Work description:

- Started to build a basic database following steps on YouTube and online websites.
- Finishing the final stages of the application being built.
- Started working on the presentation
- Started adding information to the final report

Date: August 12th, 2021
Meeting Time: 9:00PM – 12:00am
Attendees: ALL

Work description:

- Finishing up the application
- Continuing our work on the database
- Continued working on the presentation
- Continued adding to the final report

Date: August 13th, 2021
Meeting Time: 9:00PM – 12:00am
Attendees: ALL

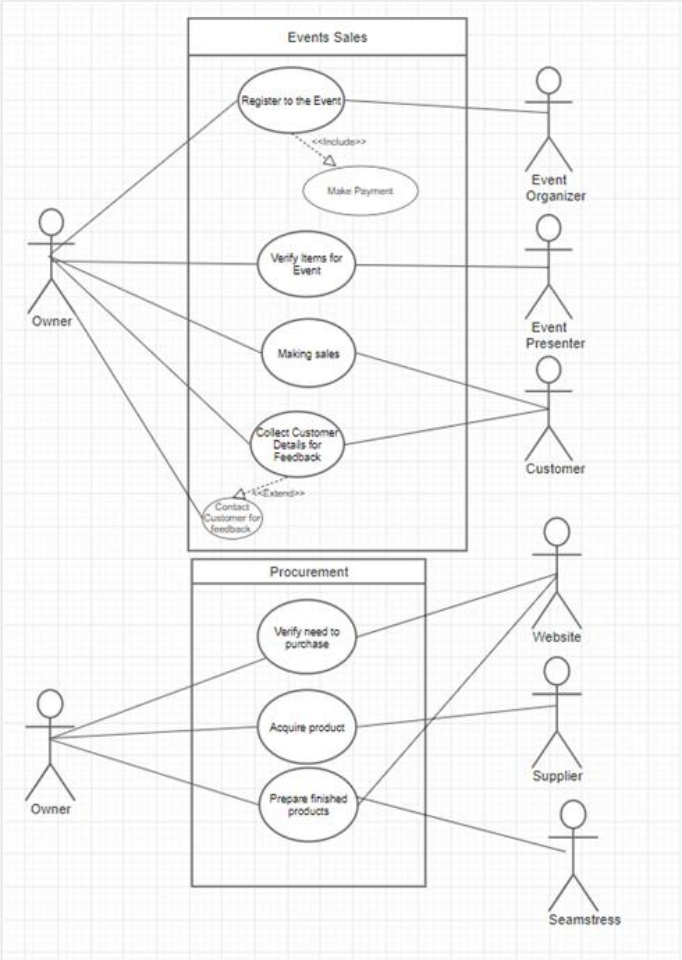
Work description:

- Adding some final touches to the application
- Ran the small data base with the application, some errors occurred which needed to be addressed.
- Finishing up the work on the presentation
- Continued adding to the final report

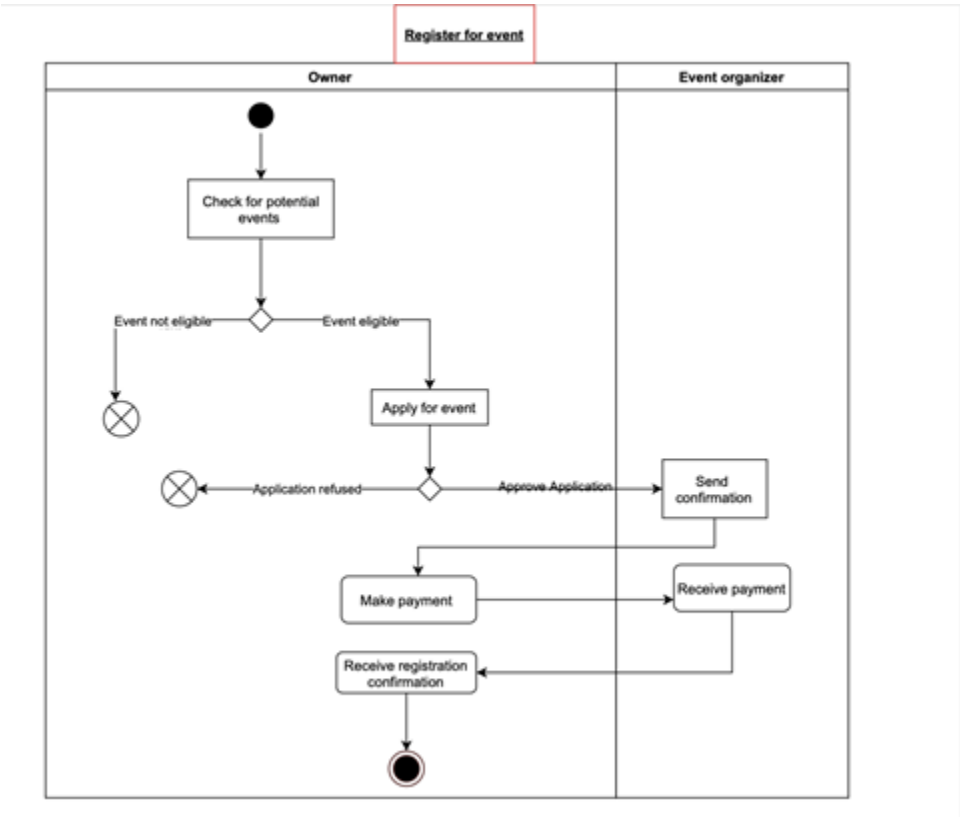
Date: August 13th, 2021
Meeting Time: 9:00PM – 12:00am
Attendees: ALL
Work description:

- Finishing up the demo
- Finishing up the Data base making sure everything runs smoothly
- Practice the presentation (debating to record or go live)
- Continued adding to the final report

APPENDIX 4: OLD VERSION OF USE CASE DIAGRAM AND USE CASE DESCRIPTION



APPENDIX 5: OLD VERSION OF THE ACTIVITY DIAGRAM AND USE CASE DESCRIPTION



Use Case Name: Register to the event	ID: 1.01	**Importance** Level: High
Primary Actor: Owner	**Use Case Type: Detail**, Essential	
Stakeholders and Interests: Owner, wants to register to an event Event organizer, wants to approve event		
Brief Description: This use case describes the different steps that are necessary to register for an event.		
Trigger: The owner wants to register to an event. **Type:** External		

Relationships:
Association: Owner, Event organizer
Include: Make payment
Extend: -
Generalization: -

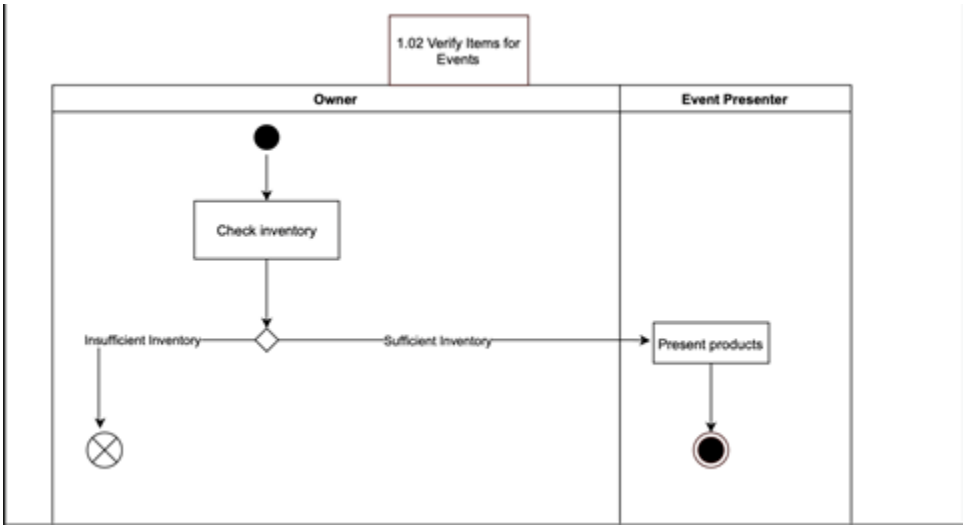
Normal Flow of Events:

1. The owner checks for potential events
   If she doesn't find an eligible event, then the process ends
   If she finds an eligible event, then the process goes to step 2
2. The owner applies for the event
   If the application is refused, then the process ends
   If the application is approved, then the process goes to step 3
3. The event organizer sends confirmation
4. The owner makes payment
5. Then, the event organizer receives payment
6. The owner receives registration confirmation
7. End

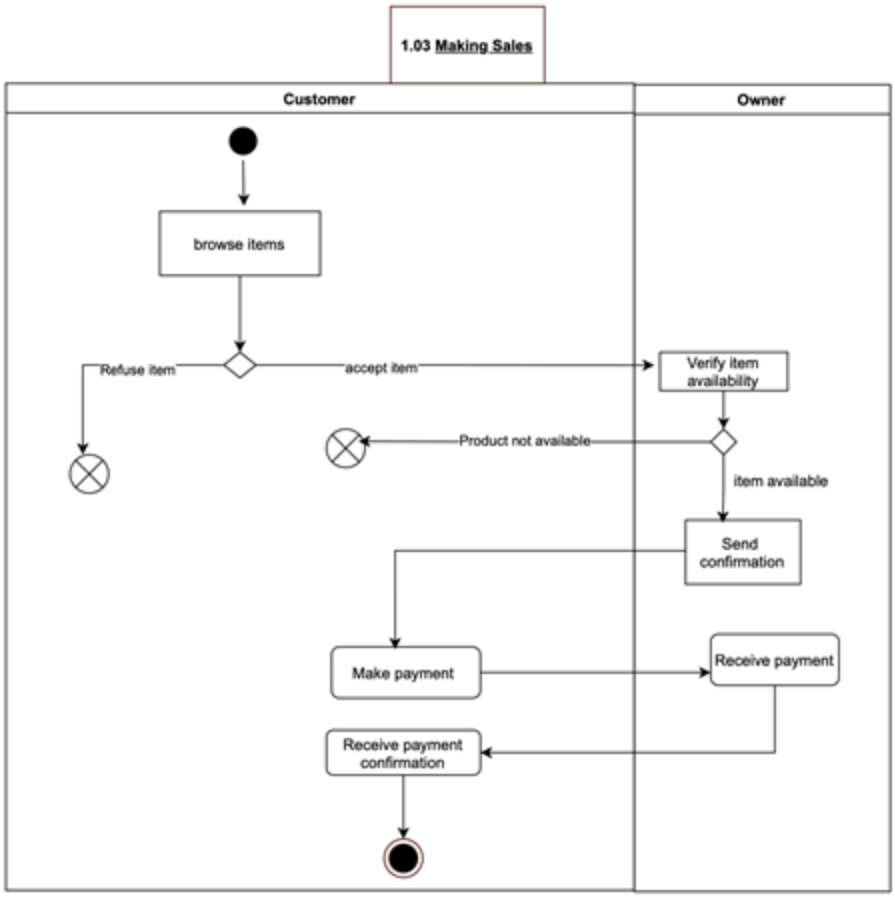
SubFlows:

S-1

Alternate/Exceptional Flows:

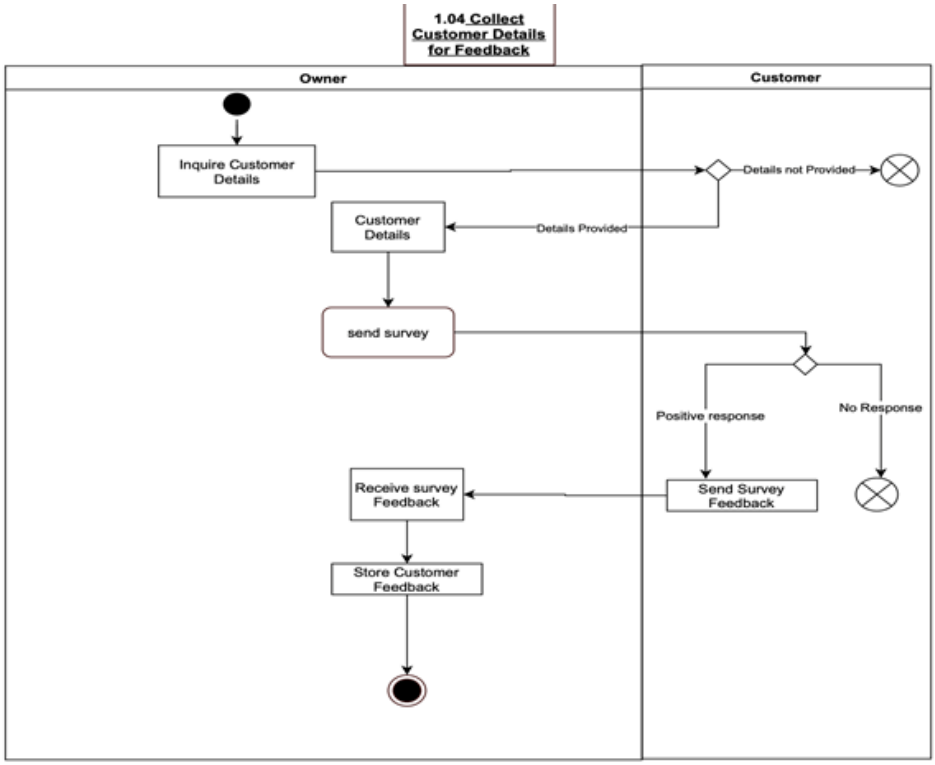


Use Case Name: Verify items for event	**ID:** 1.02	**Importance Level:** High
Primary Actor: Owner	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Owner needs to verify items in the inventory Event presenter, must present products		
Brief Description:This use case describes the different steps that are necessary to verify the inventory.		
Trigger: The owner wants to verify the inventory **Type**: Temporal		
Relationships: **Association**: Owner, Event presenter **Include: -** **Extend: -** **Generalization: -**		
Normal Flow of Events: 1. The owner checks inventory If the inventory is insufficient, then the process ends If the inventory is sufficient, then the process goes to step 2 2. The event presenter introduces the products 3. End		
SubFlows: S-1:		
Alternate/Exceptional Flows:		



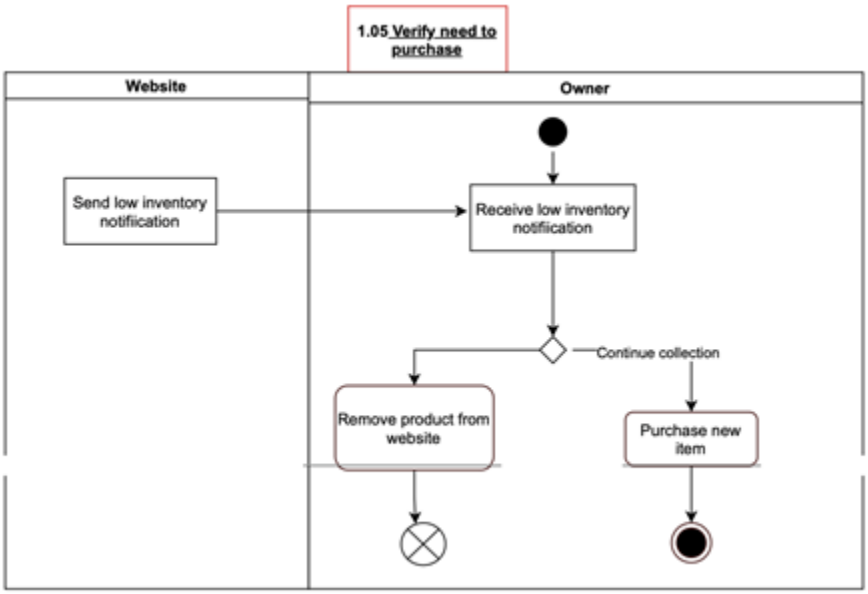
Use Case Name:Making Sales	**ID:** 1.03	**Importance Level:** High
Primary Actor: Customer	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Customer wants to purchase an item Owner, wants to make a sale		
Brief Description: This use case describes the different steps that are necessary to make a sale.		
Trigger: The customer wants to purchase an item **Type: External**		

Relationships: Association:Customer, Owner Include: Extend: Generalization:
Normal Flow of Events: 1. Customer browse inventory If item is refused, the process ends If the item is accepted, then the process goes to step 2 2. Owner verifies item availability If item is not available, then the process ends If an item is available, then the process moves to step 3 3. Owner sends order confirmation to customer 4. Customer makes payment 5. Owner receives payment 6. Customer receives payment confirmation 7. End
SubFlows:
Alternate/Exceptional Flows:



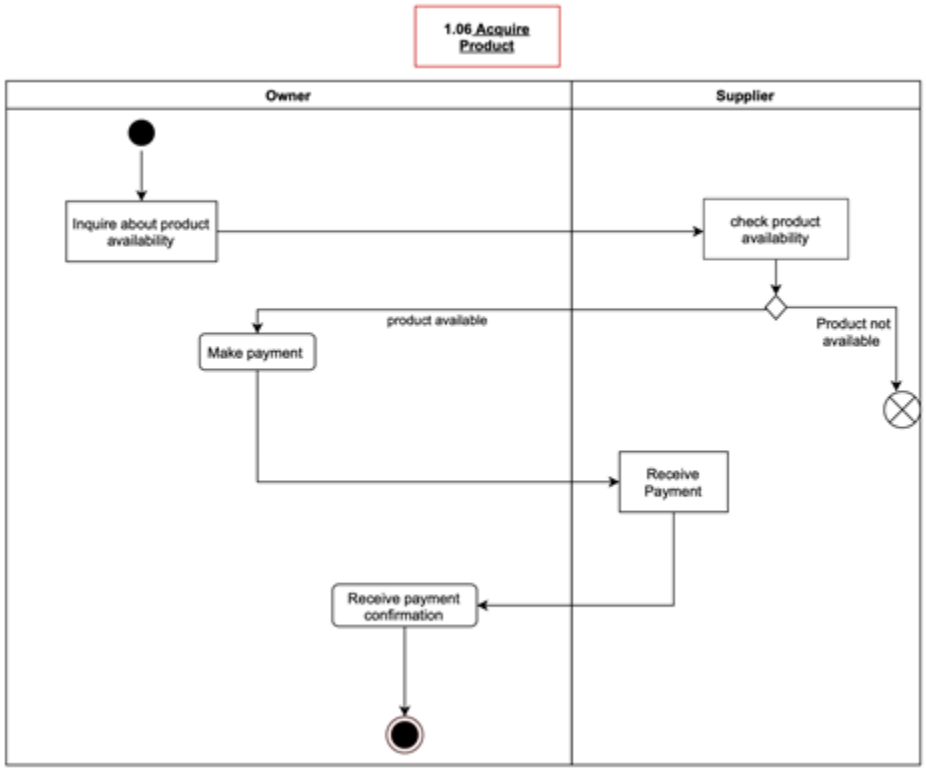
Use Case Name: Collect customer details for feedback	**ID:** 1.04	**Importance Level:** High
Primary Actor: Owner	**Use Case Type: Detail**, Essential	
Stakeholders and Interests: Owner, wants to collect customer details for feedback Customer wants to send feedback		
Brief Description: This use case describes the different steps that the owner needs to collect customer details.		
Trigger: The owner wants to collect customer details **Type: Temporal**		
Relationships: Association: Owner, customer Include: - Extend: Contact customer for feedback Generalization: -		
Normal Flow of Events: 1. The owner inquires customer details ➢ If the details are not provided, the process ends ➢ If the details are provided, the process goes to step 2 2. The owner receives customer details 3. The owner sends the survey to the customer ➢ If there is no response, then the process ends ➢ If there is a response, then the process goes to step 4 4. The customer sends the survey feedback 5. The owner receives feedback 6. The owner stores the feedback 7. End		
SubFlows: S-1:		

Alternate/Exceptional Flows:



Use Case Name: Verify need to purchase	**ID:** 1.05	**Importance Level:** High
Primary Actor: Owner	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Owner, wants to restock inventory Website, wants to update its inventory list		
Brief Description: This use case describes the different steps that the owner needs to restock inventory		
Trigger: the owner wants to restock inventory **Type:**External		

Relationships: Association: Owner, Website Include: - Extend: - Generalization: -
Normal Flow of Events: 1. Website sends owner a notification of low inventory stock 2. Owner receives low inventory notification ➢ If the product is discontinued, then the process goes to step 3 ➢ If the product is still in the collection, then the process goes to step 4 3. Owner removes product from website, and process ends 4. Owner purchases new items to restock inventory 5. End
SubFlows: S-1:
Alternate/Exceptional Flows:



Use Case Name: Acquire product	**ID:** 1.06	**Importance Level:** High
Primary Actor: Owner	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:

- The owner needs to acquire new products.
- Supplier needs to sell products.

Brief Description:This use case describes the different steps required to acquire new products

Trigger: The owner needs to acquire a new product

Type: Temporal

Relationships:

Association: Owner, Event presenter

Include: -

Extend: -

Generalization: -

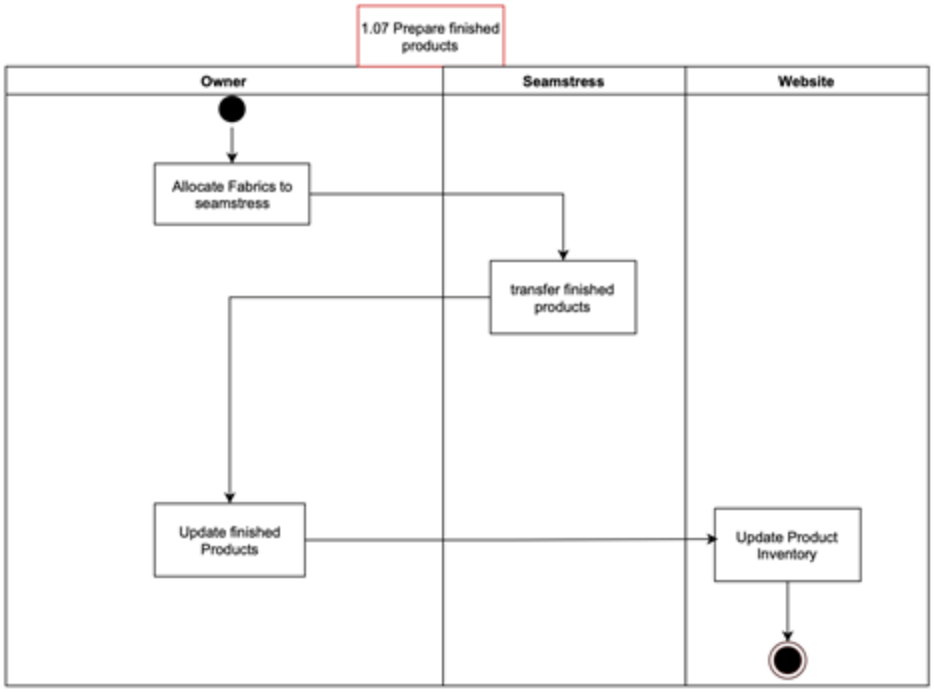
Normal Flow of Events:

1. The owner inquiries about product availability
2. Supplier checks product availability
   - If the product is not available, the process ends.
   - If product is available, the process goes to step 3
3. Owner makes payment
4. Supplier receives payment
5. Owner receives payment confirmation
6. End

SubFlows:

S-1:

Alternate/Exceptional Flows:

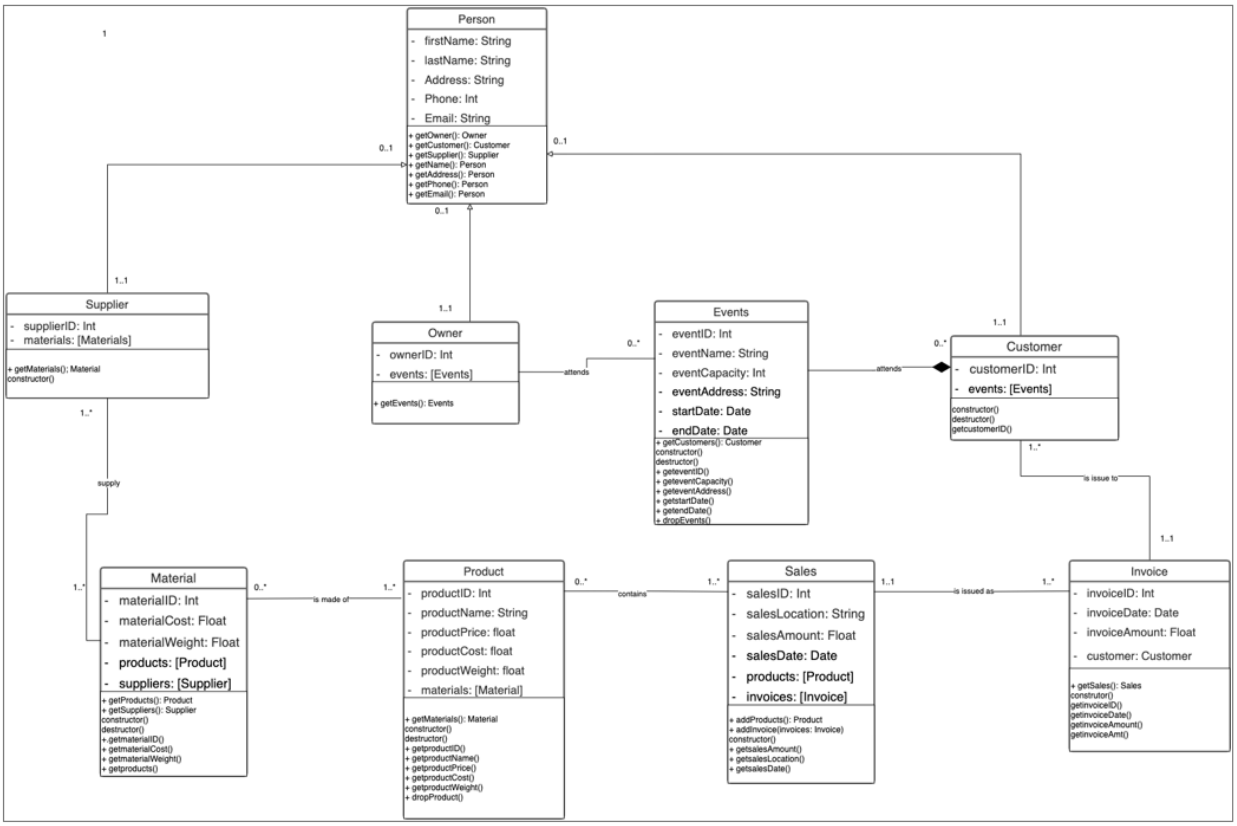


Use Case Name: Prepare finished products	**ID:** 1.07	**Importance Level:** High
Primary Actor: Owner	**Use Case Type:**Detail, Essential	
Stakeholders and Interests: Owner, wants to prepare finished goods Seamstress, wants to create new products Website, wants to update its product list		
Brief Description: This use case describes the different steps that the owner needs to take to prepare the finished goods		
Trigger: Owner wants new products **Type:** External		
Relationships: Association: Owner, Seamstress, Website Include: - Extend: - Generalization: -		

Normal Flow of Events: 1. Owner allocates fabrics to seamstress 2. Seamstress creates finished goods 3. Owner updates products 4. Website updates product inventory list
SubFlows:
Alternate/Exceptional Flows:

APPENDIX 6: OLD VERSION OF THE CLASS DIAGRAM AND THE CRC CARDS

CLASS DIAGRAM



CRC CARDS

Front:

Class Name: Supplier	**ID:** 5	**Type:** Concrete, domain
Description: Suppliers of primary materials that are used to create products.		**Associated Use Cases:** Purchase Material
Responsibilities getMaterials(): Material – determine what suppliers supply which material / desired material constructor() – to create supplier instances		**Collaborators** - Material

Back:

Attributes: *Specialized* supplierID: Int - (primary key) materials: [Materials] - (links / forms relationship materials with supplier for quick quoting, sourcing) *Generalized* firstName: String lastName: String Address: String Phone: Int Email: String
Relationships: **Generalization (a-kind-of):** Person **Aggregation (has-parts):** - **Other Associations**:material

Front:

Class Name: Sales	**ID:** 4	**Type:** Concrete , domain
Description: Total good sold at events and website		**Associated Use Cases:** Register sales
Responsibilities addInvoice() constructor() addProducts() getsalesAmount() getsalesLocation() getsalesDate()		**Collaborators** Invoice Product

Back:

Attributes: salesID salesLocation salesAmount salesDate products: [Product] invoices: [Invoice]
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** **Other Associations:** Product, invoice

Front:

Class Name: Events	**ID:** 1	**Type:** Concrete, domain
Description: Events are the social occasions where sales are made by owners and customers		**Associated Use Cases:** Verify items for the event, Register sales
Responsibilities getCustomer()		**Collaborators**

GetEventCapacity() geteventAddress() getstartDate() getendDate() dropEvents()	Owner Customer

Back:

Attributes: eventId eventName eventCapacity eventAddress startDate endDate
Relationships: **Generalization (a-kind-of):** **Aggregation (has-parts):** **Other Associations:** Owner,

Front:

Class Name: Owner	**ID:** 3	**Type:** Concrete, domain
Description: The owner is the main actor of the system who sells at events and orders raw materials.		**Associated Use Cases:** Register to the Event, Make Payment, Verify Items for events, making sales, Collect Customer Details for Feedback, Contact Customer for feedback, Verify Need to Purchase, Acquire product, Prepare finished products.
Responsibilities getEvents(): Events – determine which event to go		**Collaborators** Events

Back:

Attributes: *Specialized* OwnerId Events *Generalized* firstName: String lastName: String Address: String Phone: Int Email: String
Relationships: **Generalization (a-kind-of):** Person **Aggregation (has-parts):** - **Other Associations:** Events

Front:

Class Name: Customer	**ID:** 2	**Type:** Concrete
Description: Customers is the class name that buys products from the owner in the event		**Associated Use Cases:** Collect customer details for feedback, Register sales
Responsibilities constructor() destructor() getcustomer()		**Collaborators** events Invoice

Back:

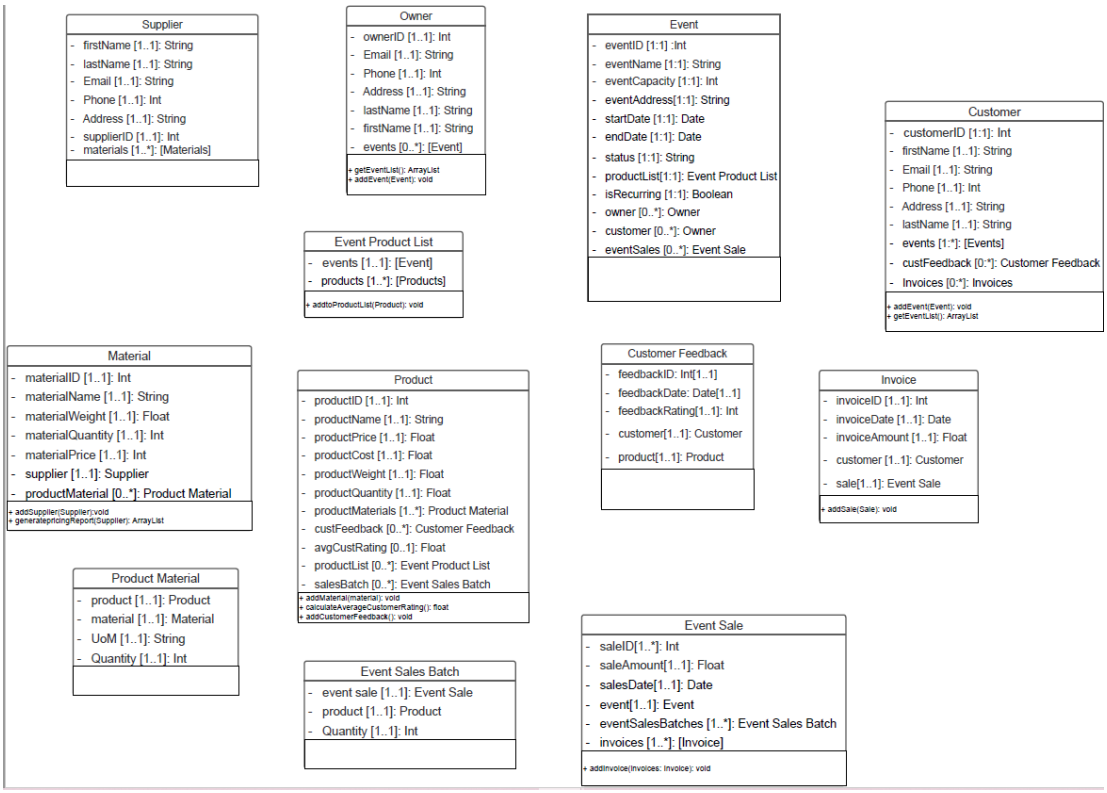
Attributes: customerID events

Relationships:
Generalization (a-kind-of): person

Aggregation (has-parts): event

Other Associations: Invoice

APPENDIX 7: OLD CLASS AND METHOD DESIGN



Procurement Narrative

Urban Wrapper is an online store with 3 employees that specializes in the creation of handmade head-wraps using African fabric. Three main processes need to be followed for the creation of the head wraps. The procurement, the production, and the sales process. For the scope of this class, the focus will be on the procurement and the sales processes. The following narrative corresponds to Urban Wrapper's procurement process.

The procurement process involves getting the raw materials needed to make the products. It all starts when Camina, the business owner, receives a notification from the website telling her that her online stock for a certain product is low. Camina first verifies the need to purchase the fabric by referring to her finished goods. If a product is low in demand, she mentally adds it onto the list of discontinued collections. If she wants to restock the same collection or procure new materials for a new line, she will make a mental note of her fabric requirements. When she has time, she will call the fabric store in St-Hubert to inquire about product availability by referring to her suppliers. Once Camina finds the fabric she needs, she verifies the quality by touch, based on her criteria. If the fabric does not meet her criteria, she chooses an alternate fabric to purchase by referring to her fabric requirements. If the supplier does not have the alternative fabric, she looks for another alternative. Once the fabric meets her criteria or she chooses an alternative, she then makes the payment to the supplier.

Subsequently, she stores her payments to her suppliers. She then receives the fabric from the supplier.

Camina has 2 employees, and she needs to share the fabric with them so that they can start the production process. When she has time, she allocates the fabric by referring to her seamstress allocation criteria, seamstress allocation quantities and owner allocation quantities. Her criteria determine how much fabric she needs to allocate to each of her employees and how much to keep to herself. Once she decides on the portion of the fabric that goes to each employee, she delivers the fabric to them.

Once the employees finish transforming the fabric into different finished goods, Camina updates her finished goods and the website with the new quantities.

Sales Narrative

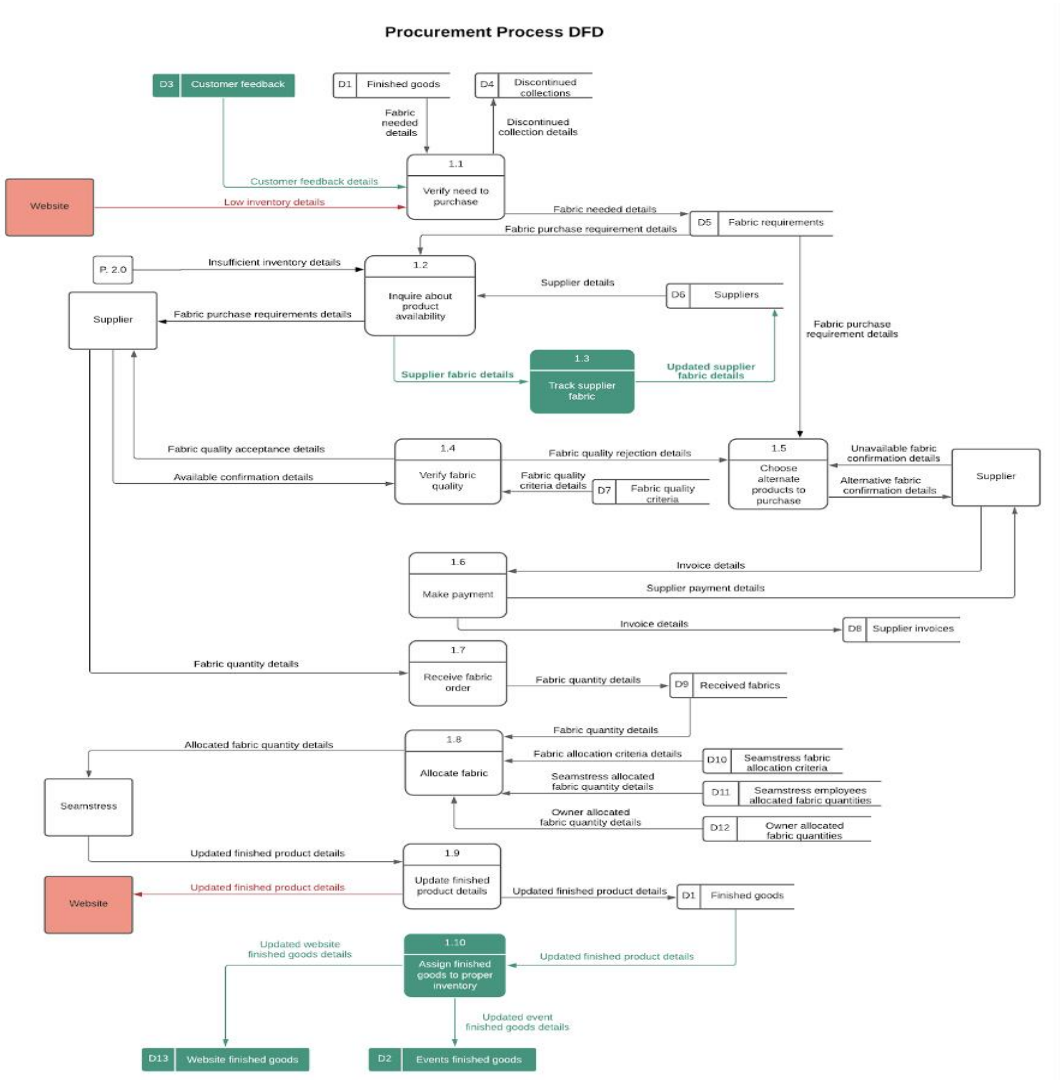
Urban Wrapper is an online store with 3 employees that specializes in the creation of handmade head-wraps using African fabric. Three main processes need to be followed for the creation of the head wraps. The procurement, the production, and the sales process. For the scope of this class, the focus will be on the procurement and sales processes. The following narrative corresponds to Urban Wrapper's sales process.

The process starts when Camina is notified of potential events that she could be interested in through an Event Notifier (mostly word of mouth or social media). She then evaluates the potential events using various criteria. Camina's criteria includes the price of the event, the location, the theme, the number of people attending and the demographic of the attendees. If the event details match her criteria for participation, she retains its relevant details in her mind, if not the process ends. When the time comes, she retrieves the event information to use it to apply for the event and forwards it to the Event Organizer. If the Event Organizer rejects her application, she goes back to evaluating other potential events. If the application is accepted, she will have to make a payment to the Event Organizer and store payment details. Once the payment is received, the Event Organizer provides the registration confirmation, which she stores for later use. When the time comes, Camina will verify the items for the event by referring to her finished goods. If she has enough products, she forwards those details to the Event presenter. However, if she does not, she must reach out to her supplier to inquire about product availability. The Event presenter is responsible for setting up the products and the table at the event.

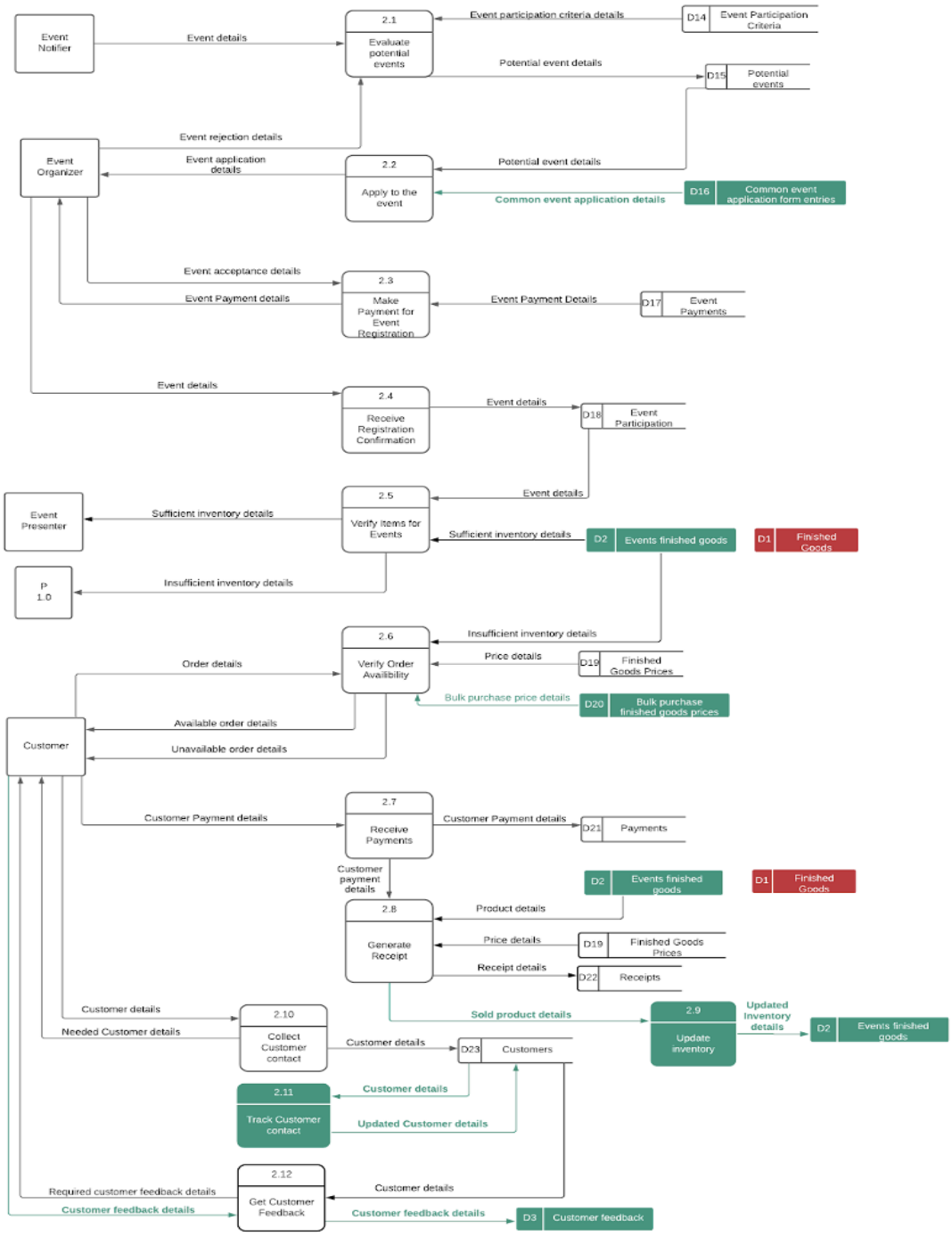
Once at the event, customers inquire about products. Camina must verify the availability of the product by referring to her finished goods and their prices. In both cases (available or unavailable), the customer is notified of the product's availability. If the product is available and the customer decides not to purchase it, or the product is unavailable the process ends. If the customer decides to purchase the product, they make a payment. Camina receives the payment and then stores the confirmation. Afterwards, a receipt is generated for the sale of which a copy is given to the customer and another copy is used to update the Finished goods data store. After which, the receipt is stored.

At the same time, Camina asks for the customers information (email and/or Instagram) in order to contact them later to get feedback or/and to invite them to future events she will be participating in. When she has time, she then reaches out to them for feedback, that is if she remembers customers information.

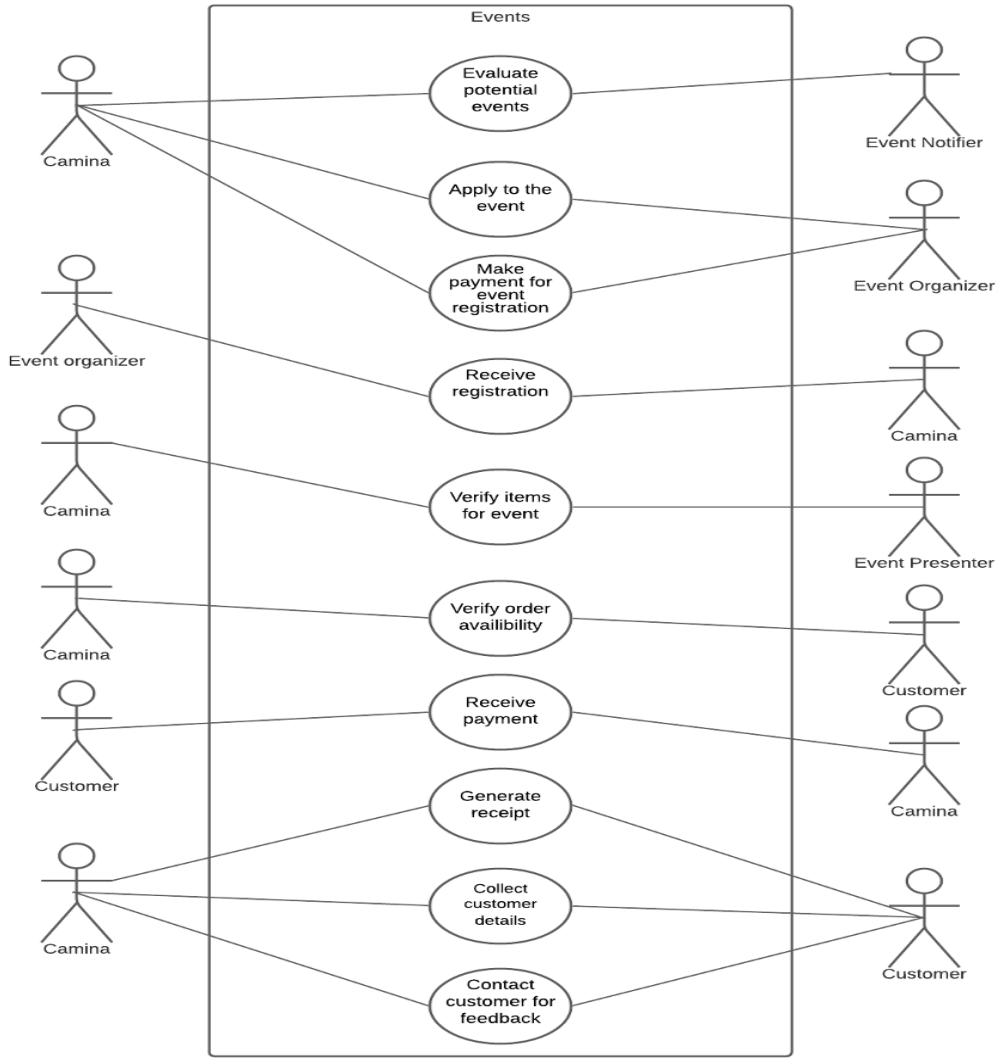
APPENDIX 9: BTM481 PROPOSED DFD'S

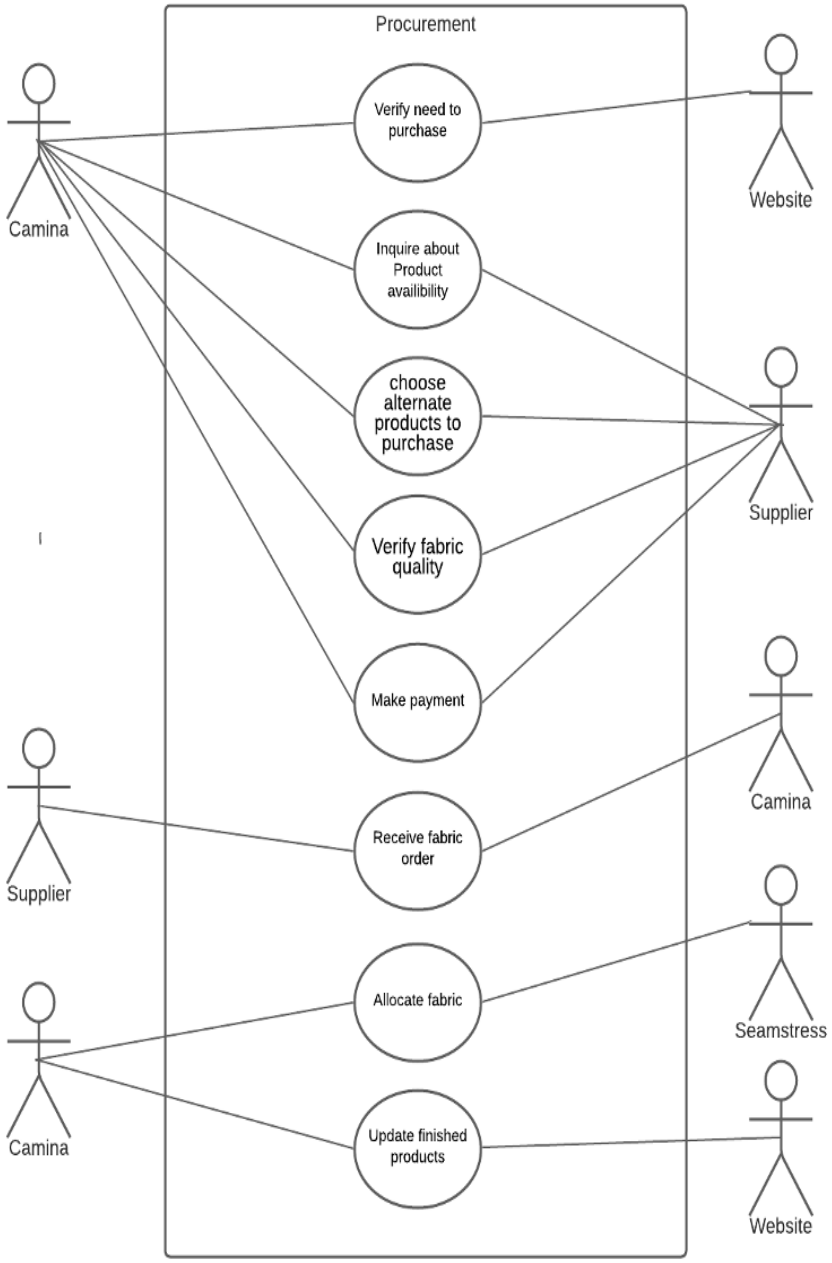


Events Sales Process DFD



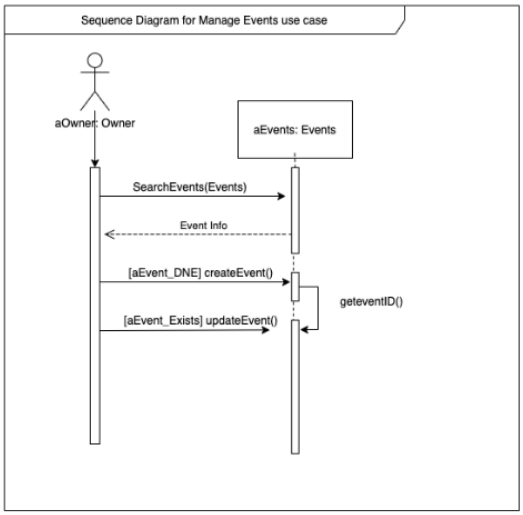
APPENDIX 10: BTM481 USES CASES

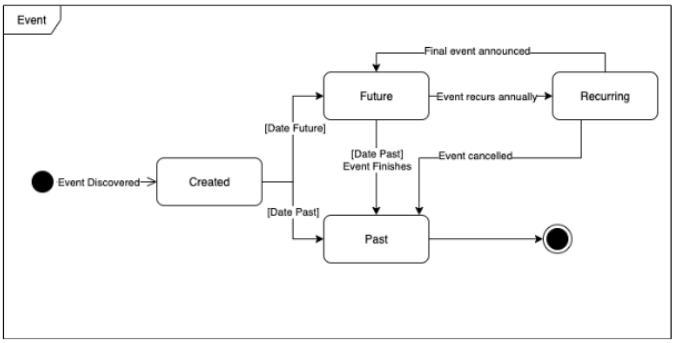



Procurement
Verify need to purchase
Inquire about Product availibility
choose alternate products to purchase
Verify fabric quality
Make payment
Receive fabric order
Allocate fabric
Update finished products
Camina
Supplier
Camina
Website
Supplier
Camina
Seamstress
Website

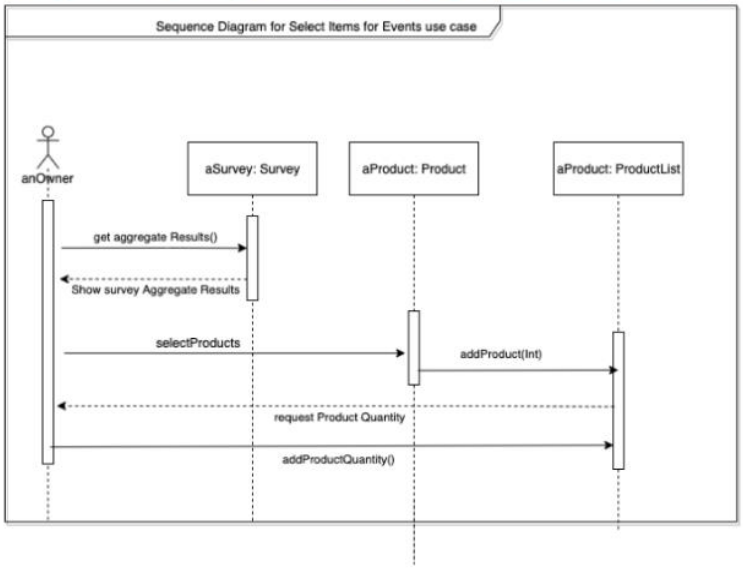
APPENDIX 11: OLD SEQUENCE DIAGRAM

Sequence Diagram & Behavioural State Machine (Manage Events):

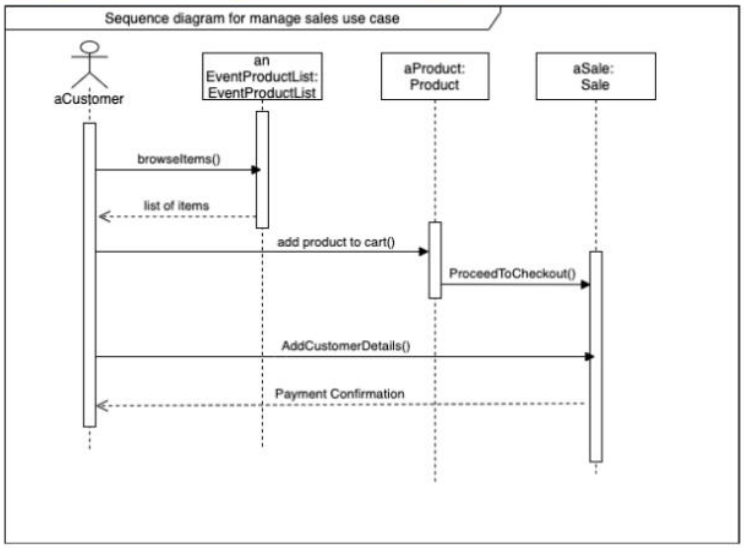




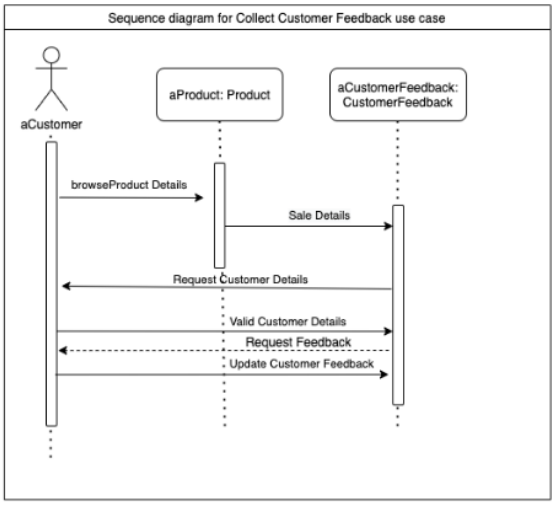
Sequence Diagram (Select Items for Event):



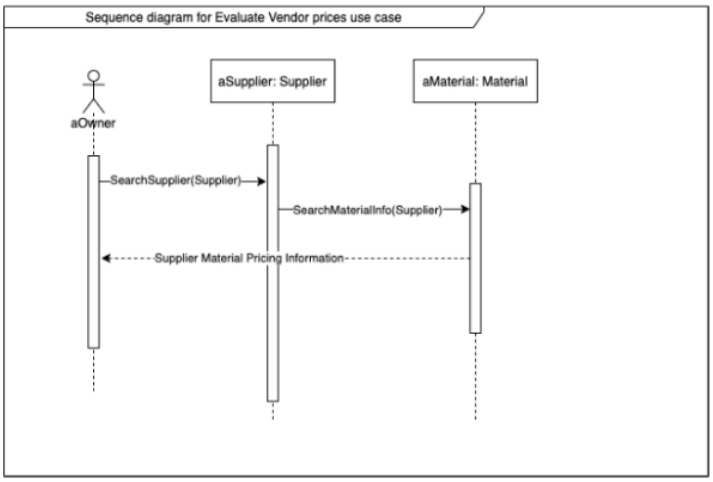
Sequence Diagram (Manage Sales):



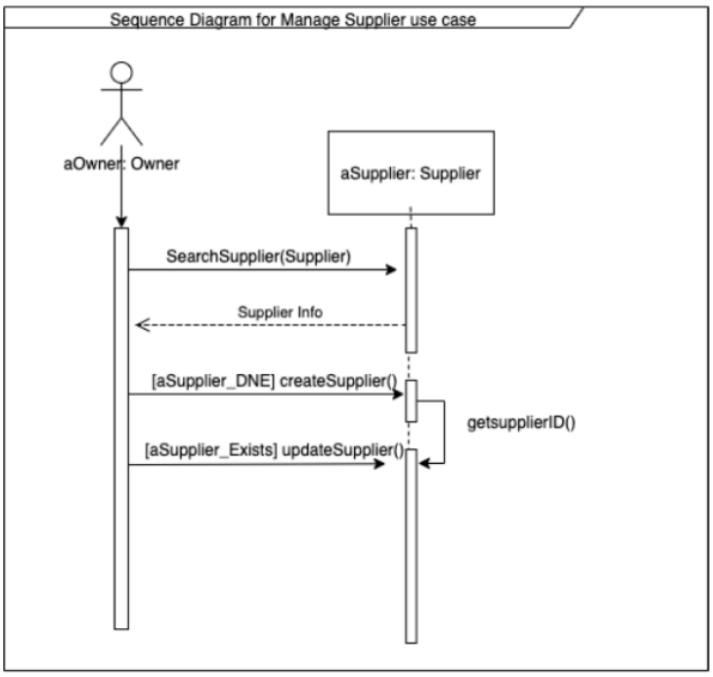
Sequence Diagram & Behavioral State Machine (Collect Customer Feedback):



Sequence Diagram (Evaluate vendor prices):



Sequence Diagram (Manage supplier):



Sequence Diagram (Manage Product):