Website URL: http://flip1.engr.oregonstate.edu:9854/

Group 34 Portfolio Assignment: Executive Summary

The first round of updates originated from feedback related to nomenclature consistency and confusion around the separate inclusion of both an “inventory” and “products” entity. As a result, we revised the names of our fields and tables to ensure that they all followed the same rules for capitalization and plurality. We also updated all of our currency fields to use the same decimal data type. Finally, additional context was added to justify the inclusion of both an “inventory” and “products” table by explaining that “inventory” was a transaction record of inventory purchases and stock levels over time.

The next round of feedback raised questions about what the relationships between our tables represent. In response, we added examples to illustrate what real life situations were reflected in each relationship and why a specific type (1:M, M:M…etc.) was chosen as result.

A basic static website using HTML and JavaScript was created for the next iteration. This did a good job of showing off the UI, but there was no database connection or sample data. This created confusion for reviewers and resulted in less actionable feedback. One important discovery was that our foreign keys were not NULLable, which was rectified to facilitate the correct behavior for insert and delete actions.

The website was converted to a flask app for the next stage and fully utilized our queries to interact with the database and display the results on the website. This led to a variety of changes once problems were discovered during implementation. Our database has a variety of calculated fields (i.e., revenue = sales * units), and there was no simple way for a user to manually update these figures based on inputs from multiple different tables. In response, we moved all of these fields out of the database and made them dynamically calculated within the read function for each page. To ensure that these fields stayed up to date, each page was altered to automatically refresh after each user interaction (create, delete, update) to grab the newest data from the database to recalculate these fields.

For the final round, we completed implementation of create and read functionalities for all remaining pages and improved the stability of the website. There were some issues with hosting our flask website on the flip servers where occasionally only the index page shows and the other pages show an internal server error. If you wait a minute or two all pages work just fine. No clear solution was found with this after contacting TA and Professor.

Project Outline

Whitney is a small business owner of The Bamboo Closet which sells tie-dyed silk fans and other tie-dyed products. Each month, she sells approximately 200 products. She also invests in social media advertising, which she believes boosts her sales. Whitney is interested in determining which products sell the best and fastest, as well as whether there is a direct correlation between the sales of certain products and the money spent on advertising them. Whitney is also interested in how often there are repeat customers. This information will enable Whitney to make more informed business decisions regarding stock quantities and the effectiveness of advertisements.

The goal of this database is to provide data for Whitney to make important business decisions. For determining which products sell the most we can look at the Sales table. To see which products sell the fastest, we can look at the date an item was put into inventory compared to the date it was sold on the sales table. For information on the effectiveness of money spent on advertising we can look at the campaigns table and look for a correlation between the sale dates and the campaign dates. For information on repeat customers, we have a calculation attribute in the customers table that calculates how many separate times a customer has made a purchase.

Database Outline

Calculated fields are green and show up on the website via read function, but aren't in the database.

- Customers: Records the details of each customer products are sold to.
    - Attributes:
        - customerID: int, autoincrement, unique, not NULL, PK
        - customerName: varchar, not NULL
        - customerEmail: varchar
        - totalRevenue (calculation based on sum of sales): decimal (19,2)
        - salesCount (calculation based on count of sales for this customer): Int
    - Relationships:
        - 1 to M with Sales: each customer can have multiple sales

- Products: Table of all available products
    - Attributes:
        - productID: int, autoincrement, unique, not NULL, PK
        - productName: varchar, not NULL
        - productPrice: Decimal (19,2), not NULL
    - Relationships:
        - 1 to M with SaleItems: intersection table to facilitate M:M relationship with Sales table
        - 1 to M with Inventory: each product can have multiple inventory entries to reflect inventory added on different days
        - 1 to M with Campaigns: intersection table to facilitate M:M relationship with Channels table
        - Channels: Influencers on social media that are paid to promote a product

- Channels: Influencers on social media that are paid to promote a product
    - Attributes:
        - channelID: int, autoincrement, unique, not NULL, PK
        - channelName: varchar, not NULL
        - channelEmail: varchar, not NULL
        - rate: decimal (19,2), not NULL (this is a daily rate)
    - Relationships:
        - 1 to M with Campaigns: intersection table to facilitate M:M relationship with Products table

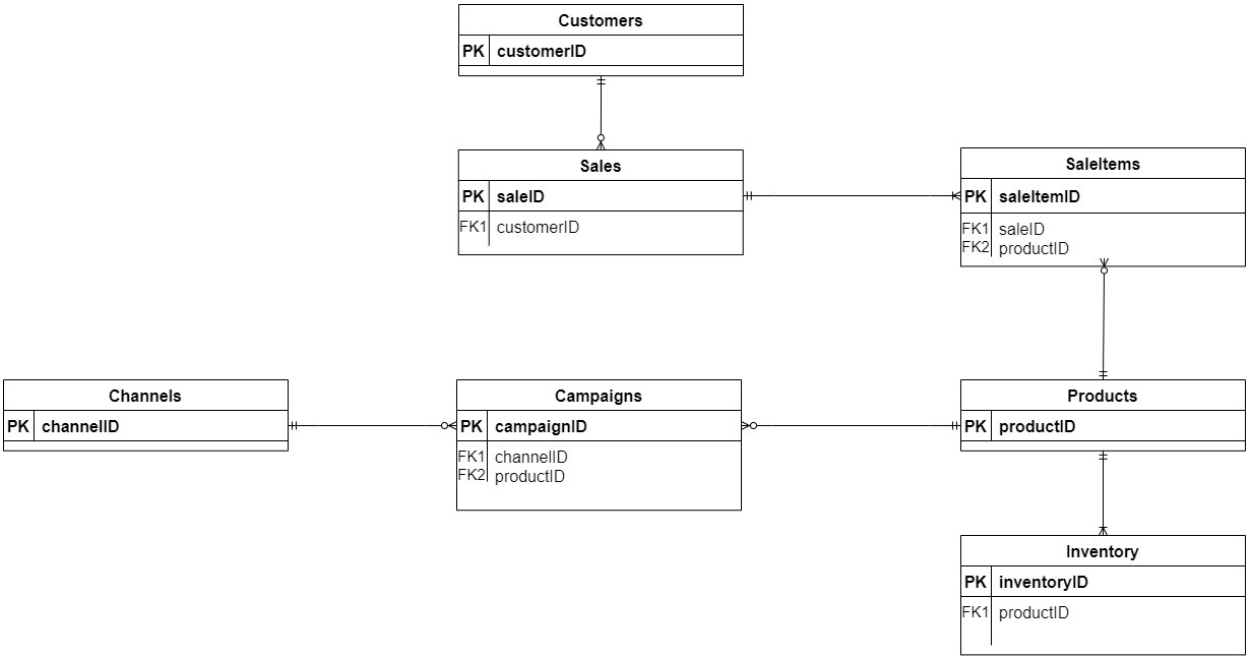
- Campaigns (intersection table):
    - Attributes:
        - campaignID: int, autoincrement, unique, not NULL, PK
        - channelID: int, FK
        - startDate: date, not NULL
        - endDate: date,
        - productID: int, FK
        - cost: decimal (19,2) (calculation based on channel.rate*length of campaign)
    - Relationships:
        - M to 1 with Channels: each channel can run multiple campaigns
        - M to 1 with Products: a product can have multiple influencer campaigns

- Sales: list of all sales The Bamboo Closet has made
    - Attributes:
        - saleID: int, autoincrement, unique, not NULL, PK
        - customerID: int, FK
        - saleDate: Date, not NULL
        - totalSaleValue (calculation based on sum of lineItemCost from saleItem): decimal (19,2)
    - Relationships:
        - M to 1 with Customers: each customer can have multiple sales
        - 1 to M with SaleItems: intersection table to facilitate M:M relationship with products

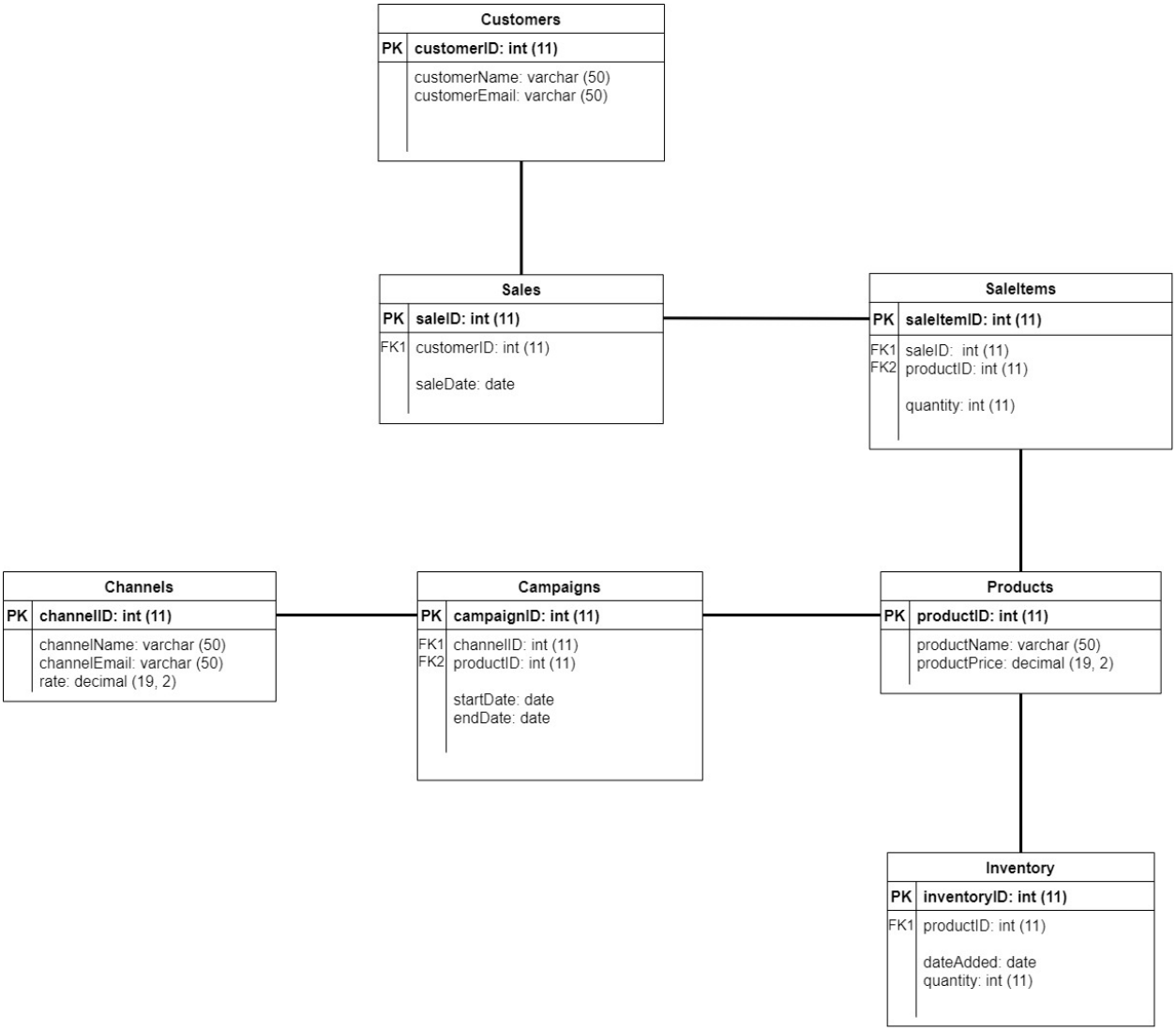
- SaleItems (intersection table): details of an individual sale
    - Attributes:
        - saleItemID: int, autoincrement, unique, not NULL, PK
        - saleID: int, FK
        - productID: int, FK
        - quantity: int, not NULL
        - totalLineItemCost (calculation of productID.productPrice * saleQuantity): decimal (19,2)
    - Relationships:
        - M to 1 with Products: a product can be associated with multiple SaleItems
        - M to 1 with Sales: a sale can have multiple SalesItems

- Inventory: list of products that are currently available for sale and their quantities
    - Attributes:
        - inventoryID: int, autoincrement, unique, not NULL, PK
        - productID: int, FK
        - dateAdded: date, not NULL
        - quantity: int, not NULL
        - totalValue (calculation based on quantity * productID.productPrice: decimal (19,2)
    - Relationships
        - 1 to M with Products: A product can have multiple inventory entries to reflect stock added on multiple different days

ERD


Customers
PK customerID
Sales
PK saleID
FK1 customerID
SaleItems
PK saleItemID
FK1 saleID
FK2 productID
Channels
PK channelID
Campaigns
PK campaignID
FK1 channelID
FK2 productID
Products
PK productID
Inventory
PK inventoryID
FK1 productID

Schema


Customers
PK customerID: int (11)
customerName: varchar (50)
customerEmail: varchar (50)
Sales
PK saleID: int (11)
FK1 customerID: int (11)
saleDate: date
SaleItems
PK saleItemID: int (11)
FK1 saleID: int (11)
FK2 productID: int (11)
quantity: int (11)
Channels
PK channelID: int (11)
channelName: varchar (50)
channelEmail: varchar (50)
rate: decimal (19, 2)
Campaigns
PK campaignID: int (11)
FK1 channelID: int (11)
FK2 productID: int (11)
startDate: date
endDate: date
Products
PK productID: int (11)
productName: varchar (50)
productPrice: decimal (19, 2)
Inventory
PK inventoryID: int (11)
FK1 productID: int (11)
dateAdded: date
quantity: int (11)

Sample Data

Green headers are calculated fields that appear on the website via read functions but are not actually inside the database.

Customers				
CustomerID	customerName	customerEmail	totalRevenue	salesCount
1	Bill	bill@gmail.com	$100.00	2
2	Dale	dale@yahoo.com.	$500000.00	250
3	Hank	hank@outlook.com	$0.00	0

Products		
productID	ProductName	ProductPrice
1	Fan	$20.00
2	Scarf	$100.00
3	Tie	$40.00

Sales			
saleID	customerID(FK)	saleDate	totalSaleValue
1	1	5/15/2023	$10000.00
2	1	6/1/2023	$15000.00
3	2	2/10/2023	$20000.00
4	2	4/5/2023	$40000.00

SaleItems				
saleItemID	saleID(FK)	productID(FK)	quantity	totalLineItemCost
1	1	2	50	$5000.00
2	1	1	250	$5000.00
3	2	2	600	$60000.00
4	3	3	1000	$40000.00

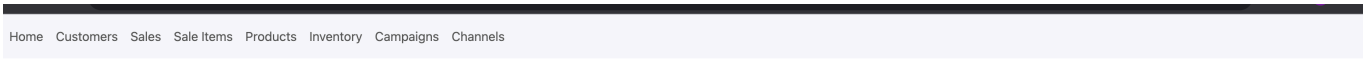
Inventory				
inventoryID	productID(FK)	dateAdded	quantity	totalValue
1	1	7/10/2023	500	$10000.00
2	1	7/16/2023	750	$15000.00
3	2	5/10/2023	200	$20000.00
4	2	6/5/2023	400	$40000.00
5	3	7/1/2023	300	$12000.00

Campaigns					
campaignID	channelID(FK)	productID(FK)	startDate	endDate	cost
1	1	2	5/15/2023	6/30/2023	$5000.00
2	1	3	6/1/2023	6/15/2023	$15000.00
3	2	1	2/1/2023	2/28/2023	$5000.00
4	2	3	4/15/2023	4/15/2023	$2500.00

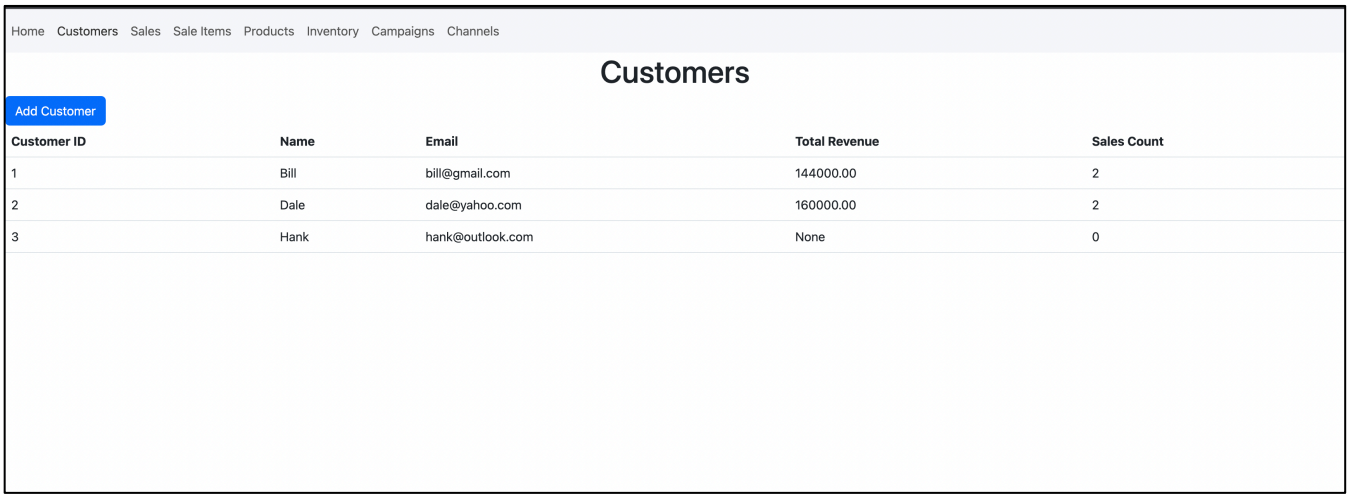
Channels			
channelID	channelName	channelEmail	channelRate
1	Koolguy	kguy@gmail.com	$1000.00
2	Supermom	s_mom@yahoo.com.	$5000.00
3	Influencer	influ@outlook.com	$10000.00

Screen Captures

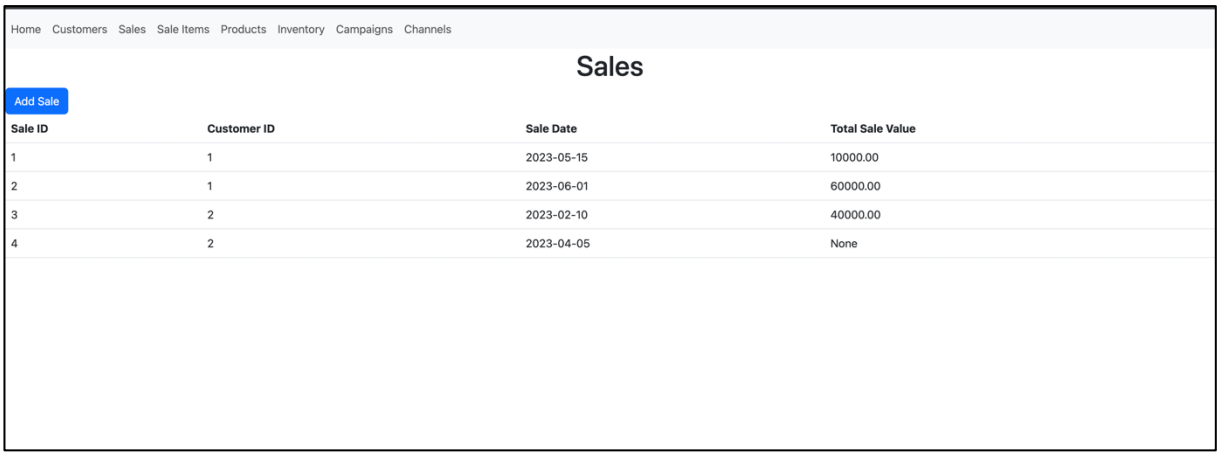
Index Page



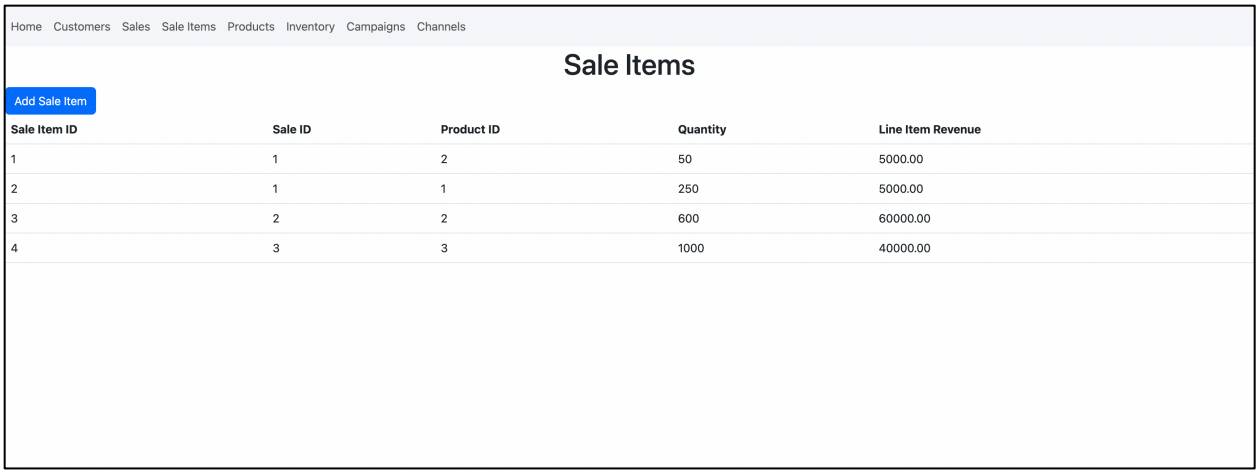
Customers Page (Create/Read)



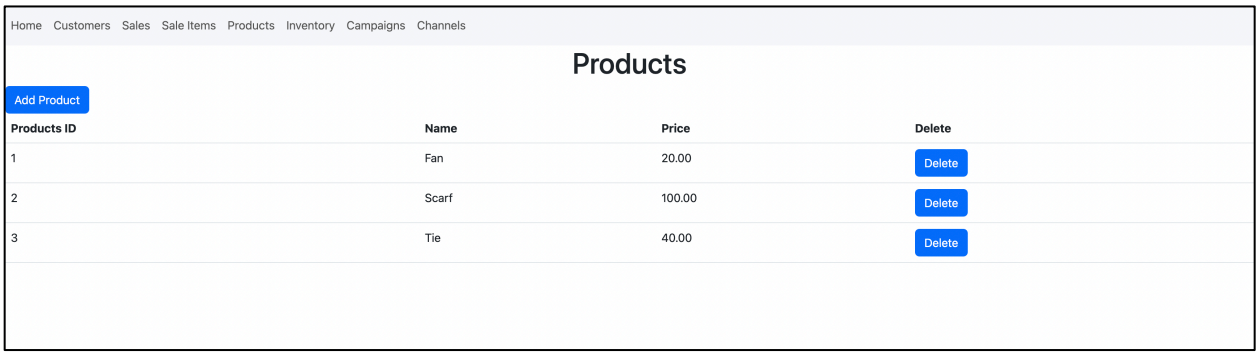
Sales Page (Create/Read



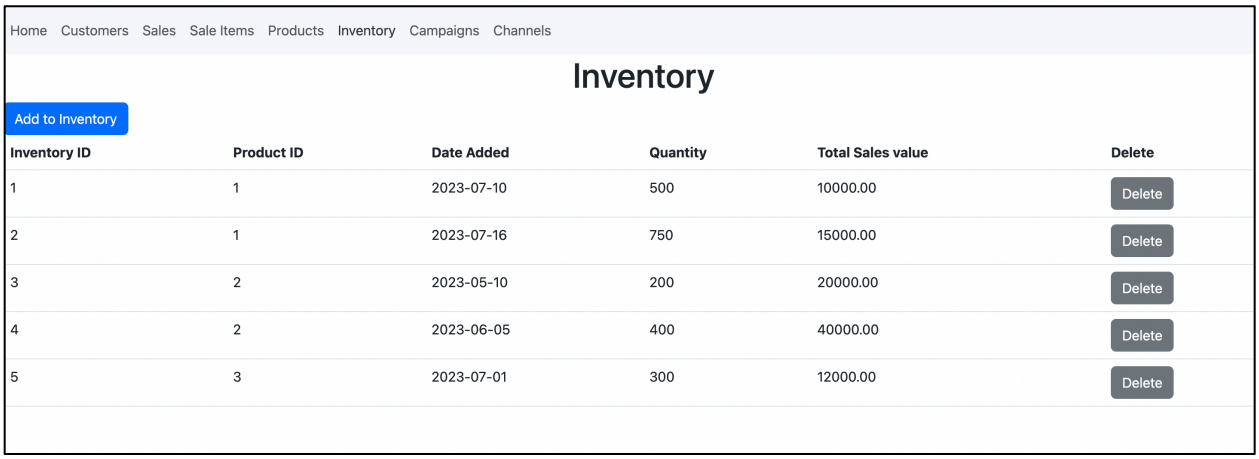
Sale Items Page (Create/Read)



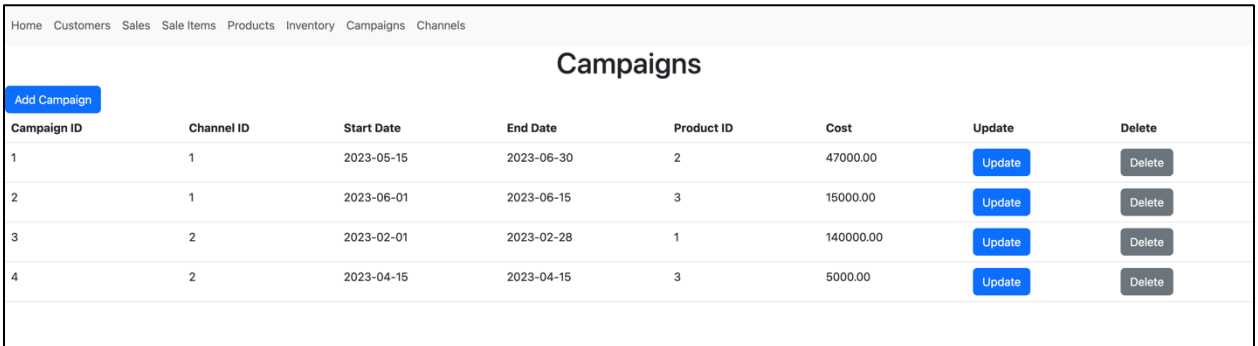
Products Page (Create/Read/Delete)



Inventory Page (Create/Read/Delete)



Campaigns Page (Create/Read/Update/Delete)



Channels Page (Create/Read/Delete)