CS 3354 Software Engineering
Final Project Deliverable 1

Storage Hub

Kuanlin Liu
Cameron Glick
Jesse Ladyman
Kevin Tran
Khiem Huynh
Yeswanth Bogireddy
Zarof Kabir

Project Description:
This software would be a storage system that would act as a complete inventory management of the stock it needs to keep track of. It would allow users to order and request items as well.

Instructor Feedback:
Please make sure that you include the following in your **final project deliverable 2 (final deliverable)**:

- A comprehensive research to find similar project implementations. Cite your findings properly using IEEE citation format.

- Make sure that you are adding extra feature(s) to uniquely differentiate your design from already existing similar implementations. Clearly explain what these feature(s) are.

- A comparison of your design with similar implementations in the field. This comparison could be presented in any format of your choice, such as a table, paragraphs, charts, etc.

Proposal:
To address these points, we will look at other softwares like Amazon and eBay to analyze the similarities and see how we can make our program more unique. The comparison will most likely be in a venn diagram. We plan to add specifications towards the warehousing aspects of the market.

GitHub Repository URL: https://github.com/KT-CSML/3354-StorageHub

Project Scope:

1.1 Inventory
  1.1.1 Stock - Number of items available.
  1.1.2 Updating Stock - Items that are received for stocking.
  1.1.3 Location of Stock in Market Place
  1.1.4 Misc Items
1.2 User Account
  1.2.1 Purchase items
  1.2.2 Request Items
  1.2.3 Refund Items
  1.2.4 Add & Delete Items
  1.2.5 Pay by Credit Card or Gift Card
  1.2.6 Pick up from Store
  1.2.7 Delivery to House
1.3 Market Location
  1.3.1 Number of Market Places
  1.3.2 Available Items
1.4 Discounts
  1.4.1 Items on Sale
  1.4.2 Employee Discount

Delegation of Tasks:

Software process model and requirements. [Kuanlin and Kevin]
Diagramming models (use case, sequence, class). [Cameron and Jesse]
Architectural design. [Khiem and Zarof]
Project scope. [Yeswanth]

Software Process Model:

The incremental process model would most likely be the best option to apply to this program as the storage system would initially be basic, with standard usage and capabilities. Each increment could work on user feedback of how to improve the inventory management or ordering system that is worked on alongside other new features to make the system more functional.

Software Requirements:

Functional:

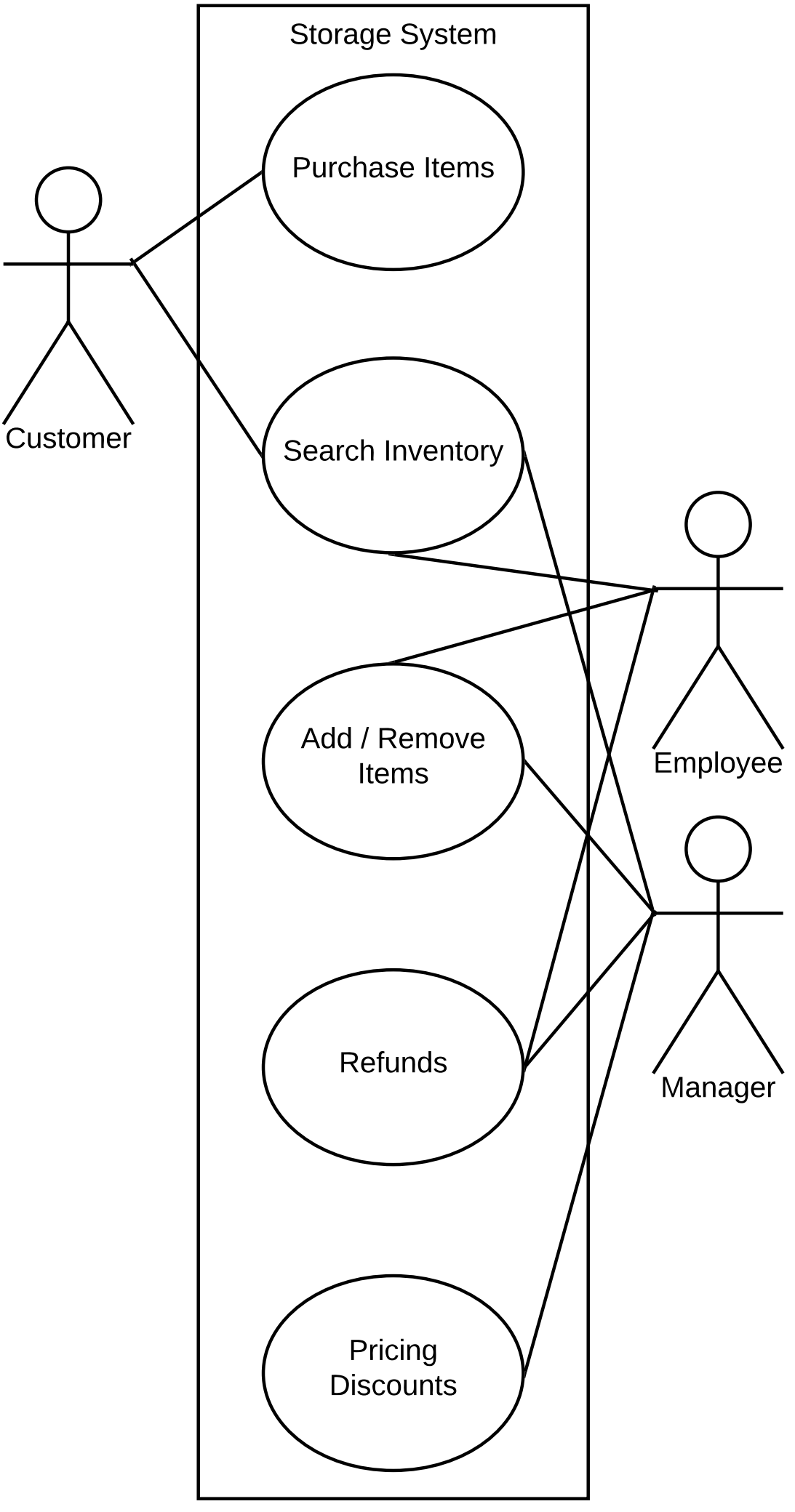
- The employee should be able to order, update and remove items in the inventory.
- The users should be able to purchase, request and refund items in the inventory.
- The users should be able to choose their payment option.
- The users should be able to choose to pick up or deliver.
- The employee should be able to decide which item has discount and have employee discount for themselves.
- The market should be able to show the available items to users and employees.

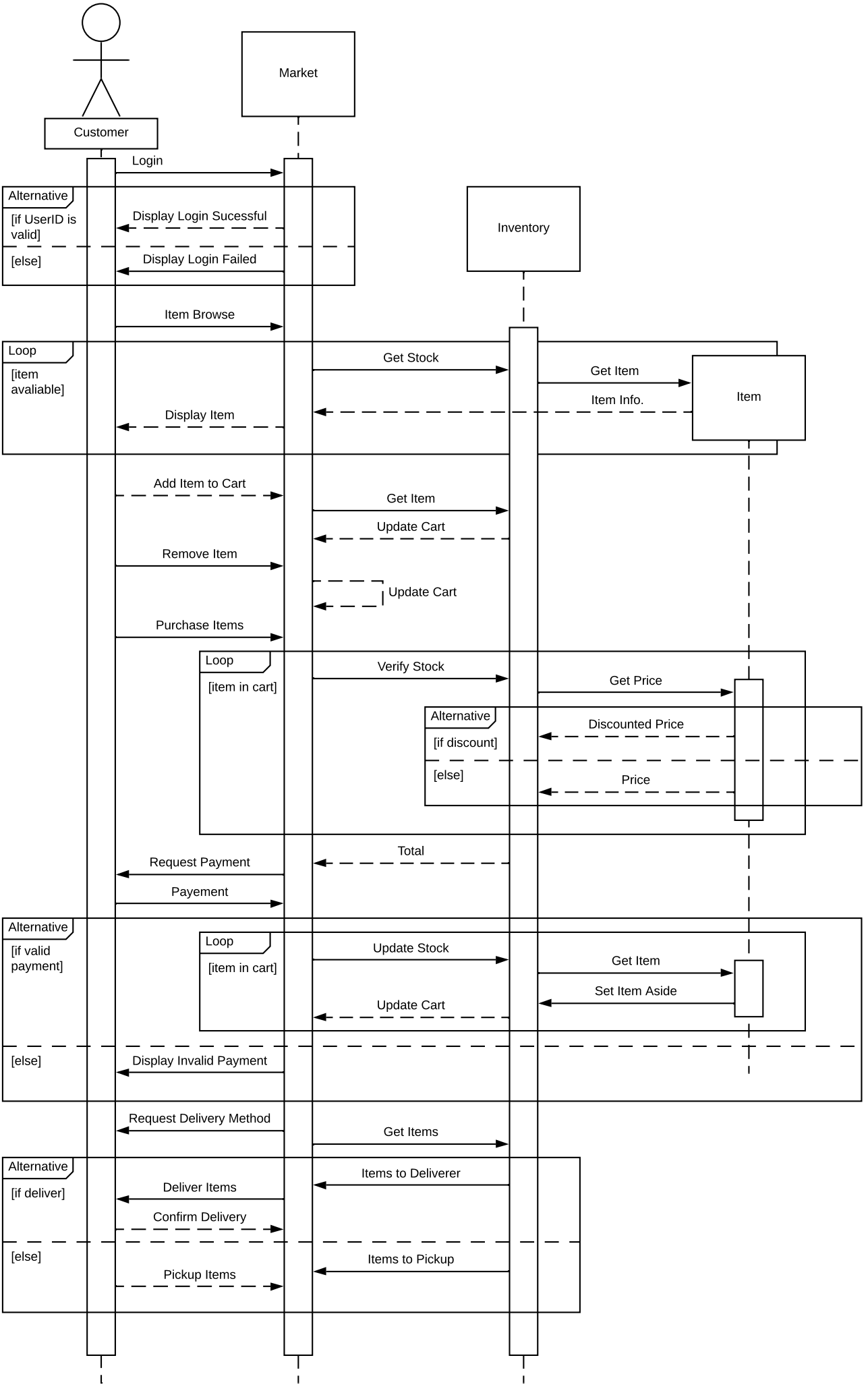
Nonfunctional:

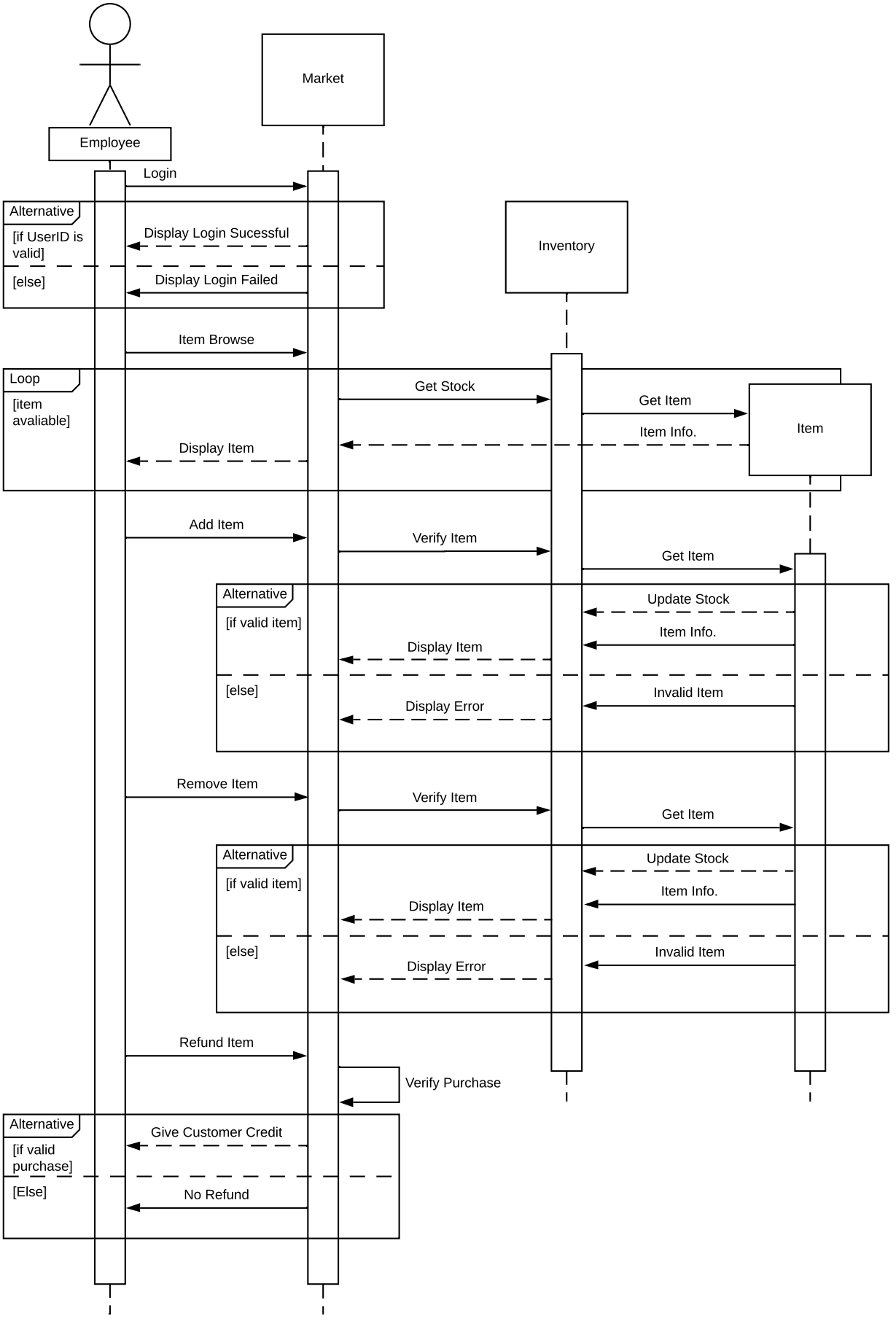
- Be able to use the storage application with response time less than 0.1 seconds.
- Have at least 10+ terabytes of storage space for data storage of inventory.
- Dependable communication between databases should be less than 5 seconds to present accurate/updated information.
- Be able to handle and support over 10,000 users concurrently.
- Systems are available at any time with maintenance frequent, lasting no longer than an hour.
- Process transactions in less than 10 seconds.
- Protect the account info of users.
- Able to connect accounts to banks and send/receive data between them.
- Ease of use with less than 5 days of training to understand the system.
- Employee accounts must be registered with a unique code provided by management.
- System operates on Windows 7, 8, and 10, macs and Linux environments.
- Development of code in C++/C#.
- Must be regulated by admins to prevent any service downtime.
- Have decently large inventories.
- Any downtime/maintenance occurs early mornings from 2-4 AM.

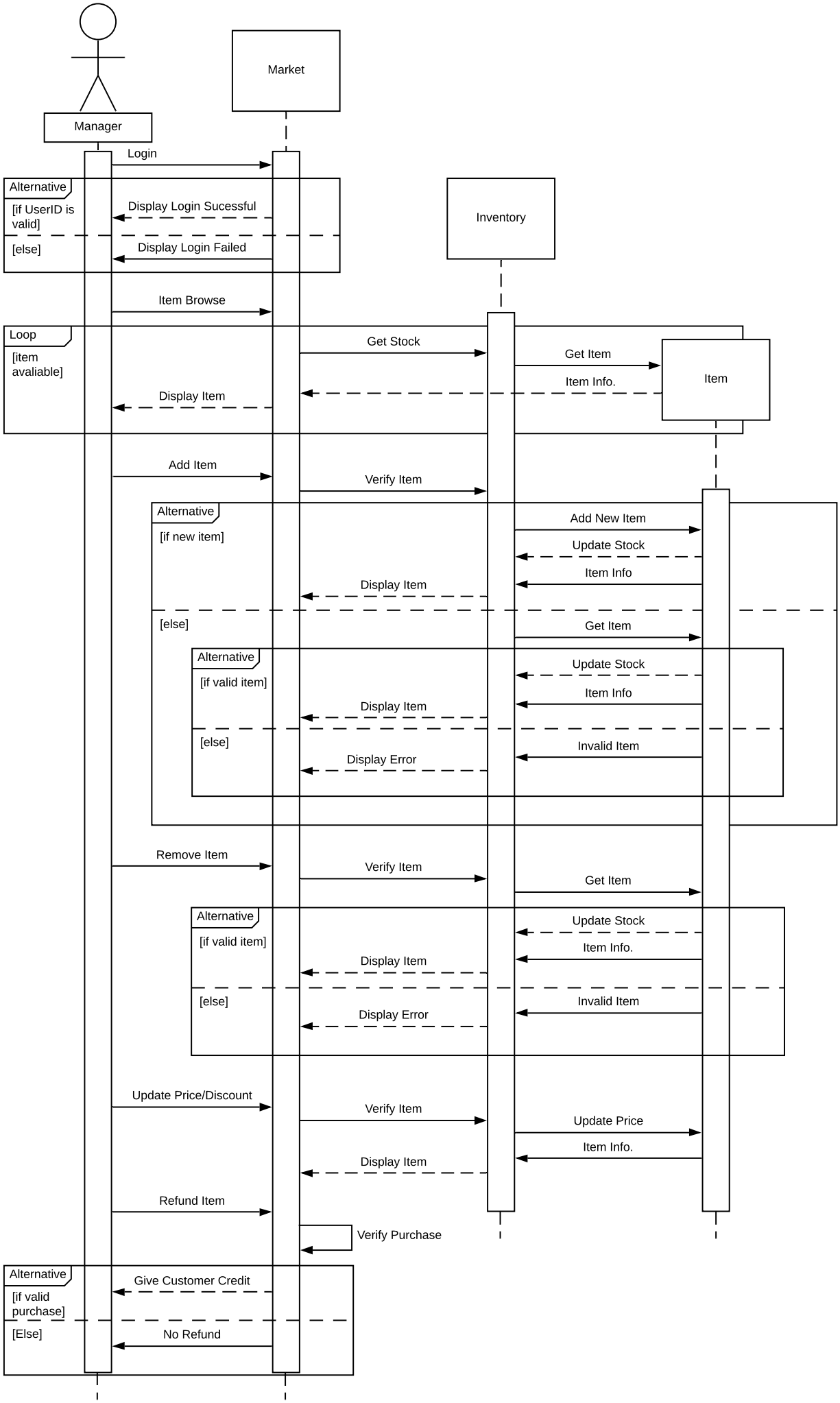
Use Case, Sequence, and Class Diagrams (+ Architectural Design as Client Server):

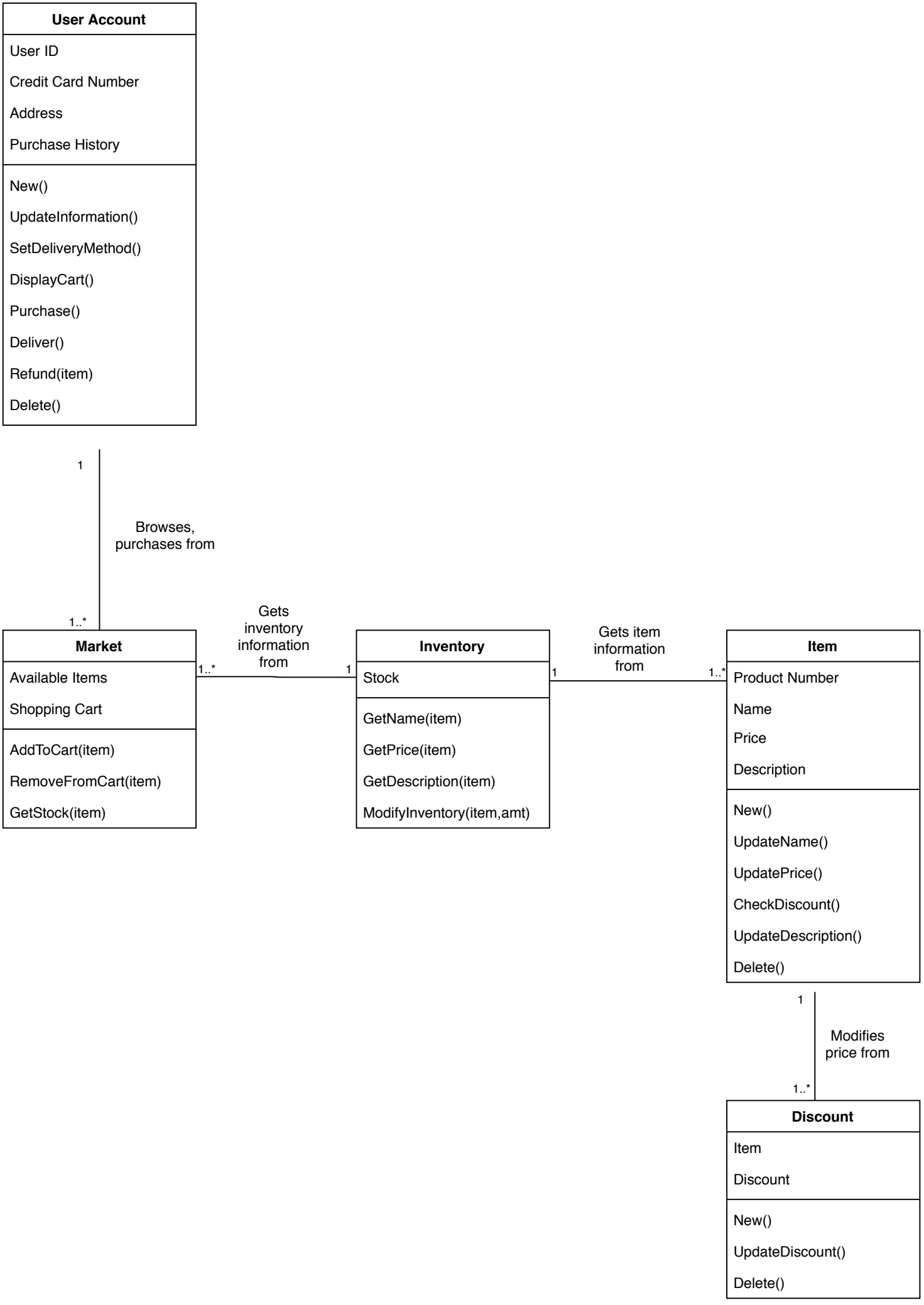
Note: *Each sequence diagram is based on the user type and contains more than one use case per sequence diagram.*

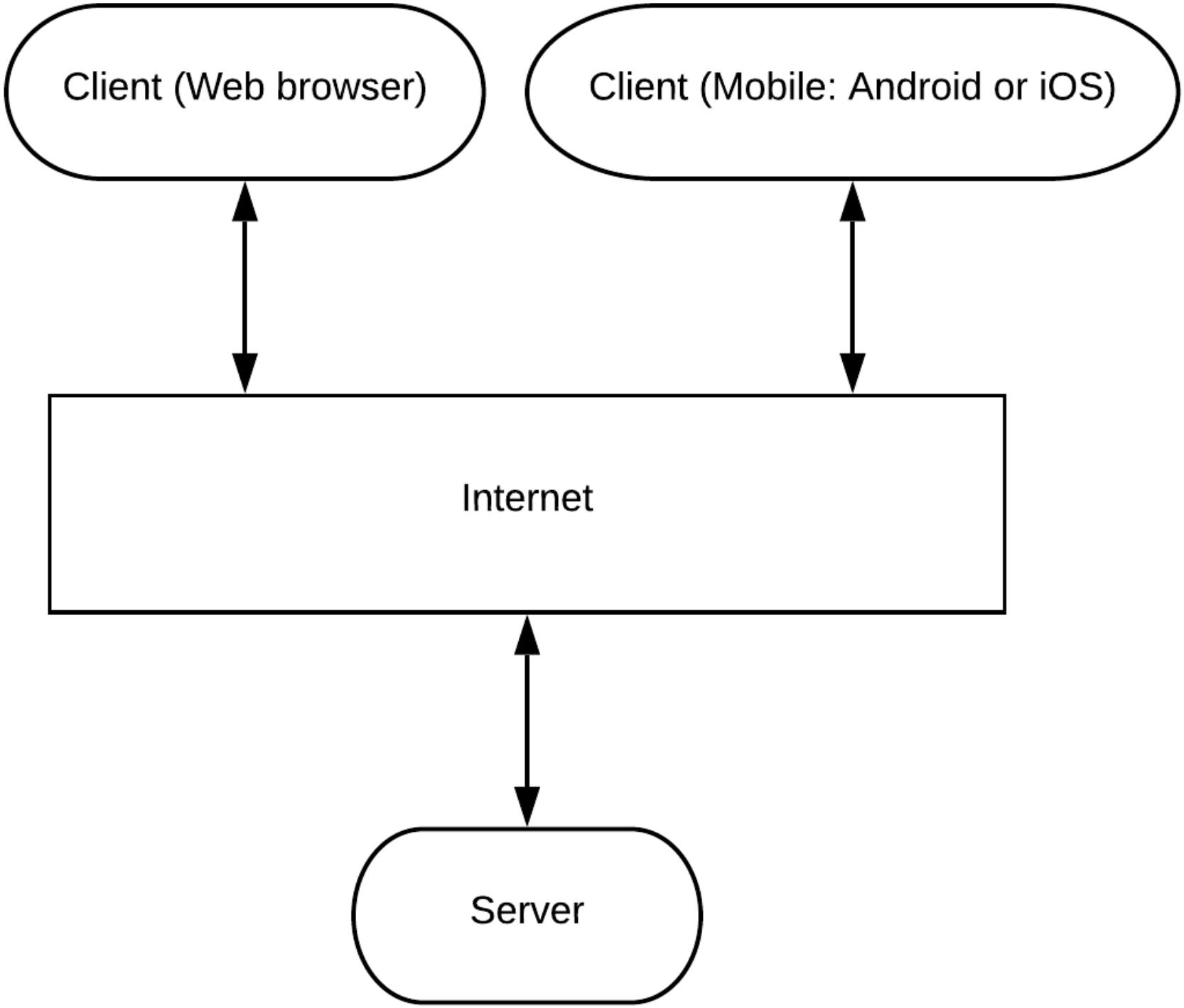

Storage System
Purchase Items
Search Inventory
Add / Remove Items
Refunds
Pricing Discounts
Customer
Employee
Manager


Customer
Market
Inventory
Item
Login
Alternative
[if UserID is valid]
Display Login Sucessful
[else]
Display Login Failed
Item Browse
Loop
[item avaliable]
Get Stock
Get Item
Item Info.
Display Item
Add Item to Cart
Get Item
Update Cart
Remove Item
Update Cart
Purchase Items
Loop
[item in cart]
Verify Stock
Get Price
Alternative
[if discount]
Discounted Price
[else]
Price
Total
Request Payment
Payement
Alternative
[if valid payment]
Loop
[item in cart]
Update Stock
Get Item
Set Item Aside
Update Cart
[else]
Display Invalid Payment
Request Delivery Method
Get Items
Alternative
[if deliver]
Items to Deliverer
Deliver Items
Confirm Delivery
[else]
Items to Pickup
Pickup Items


Employee
Market
Inventory
Item
Login
Alternative
[if UserID is valid]
Display Login Sucessful
[else]
Display Login Failed
Item Browse
Loop
[item avaliable]
Get Stock
Get Item
Item Info.
Display Item
Add Item
Verify Item
Get Item
Alternative
[if valid item]
Update Stock
Item Info.
Display Item
[else]
Invalid Item
Display Error
Remove Item
Verify Item
Get Item
Alternative
[if valid item]
Update Stock
Item Info.
Display Item
[else]
Invalid Item
Display Error
Refund Item
Verify Purchase
Alternative
[if valid purchase]
Give Customer Credit
[Else]
No Refund


Manager
Market
Inventory
Item
Login
Alternative
[if UserID is valid]
Display Login Sucessful
[else]
Display Login Failed
Item Browse
Loop
[item avaliable]
Get Stock
Get Item
Item Info.
Display Item
Add Item
Verify Item
Alternative
[if new item]
Add New Item
Update Stock
Item Info
Display Item
[else]
Get Item
Alternative
[if valid item]
Update Stock
Item Info
Display Item
[else]
Invalid Item
Display Error
Remove Item
Verify Item
Get Item
Alternative
[if valid item]
Update Stock
Item Info.
Display Item
[else]
Invalid Item
Display Error
Update Price/Discount
Verify Item
Update Price
Item Info.
Display Item
Refund Item
Verify Purchase
Alternative
[if valid purchase]
Give Customer Credit
[Else]
No Refund


User Account
User ID
Credit Card Number
Address
Purchase History
New()
UpdateInformation()
SetDeliveryMethod()
DisplayCart()
Purchase()
Deliver()
Refund(item)
Delete()
1
Browses, purchases from
1..*
Market
Available Items
Shopping Cart
AddToCart(item)
RemoveFromCart(item)
GetStock(item)
1..*
Gets inventory information from
1
Inventory
Stock
GetName(item)
GetPrice(item)
GetDescription(item)
ModifyInventory(item,amt)
1
Gets item information from
1..*
Item
Product Number
Name
Price
Description
New()
UpdateName()
UpdatePrice()
CheckDiscount()
UpdateDescription()
Delete()
1
Modifies price from
1..*
Discount
Item
Discount
New()
UpdateDiscount()
Delete()


Client (Web browser)
Client (Mobile: Android or iOS)
Internet
Server