Final Report: Digital Marketplace

Group 26
Anthony Tolentino 30081427
Arcleah Pascual 30056034
Tyler Nguyen 30158563

Introduction:

The problem that we chose to address was the trust in online peer-to-peer clothing transactions, specifically the trust aspect of the transaction, such as whether the seller would send the clothing item after receiving the money from the buyer. Our system to address this problem is to create a website that holds the inventory of sellers where buyers can browse and purchase items. However, the main difference from our system compared to other competitors is that we have an integrated middleman service that will oversee the transactions. This middleman service will first hold the funds of the buyer before sending the funds to the seller to ensure that the buyer will receive the seller's item.

Project Design:

In this system, each public user is required to register for an account, during the registration, they are prompted to pick between a seller or buyer. The middleman also has their own account which will be predefined. If the user decides to go with a seller account they will be able to list their items, during the listing they are asked for the name, condition, description, price, size, and a picture of the item. They are also in charge of shipping the item out and deleting the listing after the buyer has received the item. If the user decides to go with a buyer account, then they will be able to see all clothing items listed and place an order on the item, they will be asked to pay for the order, and these funds will go to the middleman, where it will be held until the buyer receives the order from the seller. Once the order is placed, the seller will get notified to ship the order, and when the buyer receives the order, they are prompted to notify the middleman, and then the middleman will release the funds to the seller. The middleman account is responsible for moderating the transaction between buyer and seller, but most importantly releasing the buyer funds to the seller once the buyer has received the seller's item.
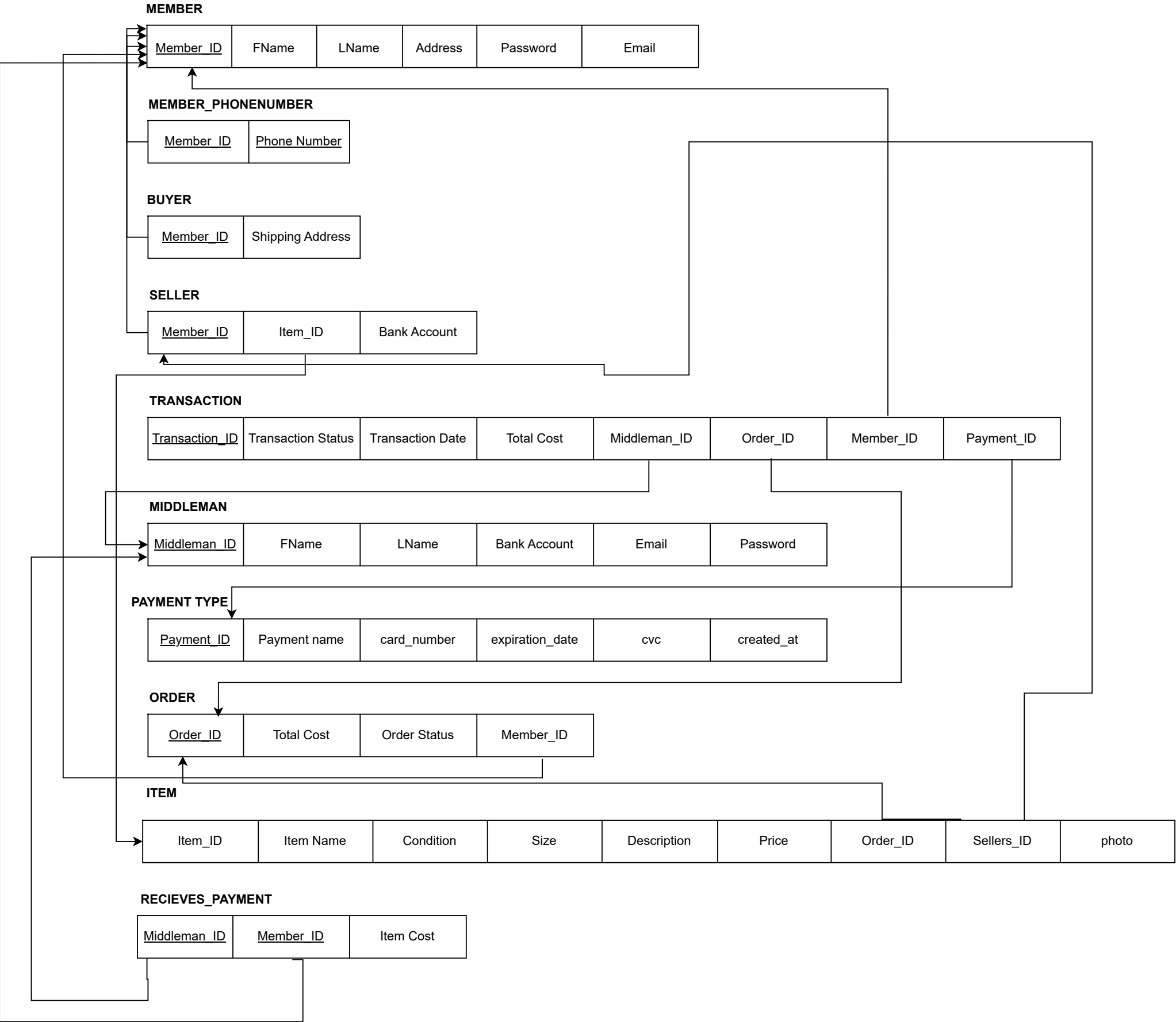
Project Design EER Diagram
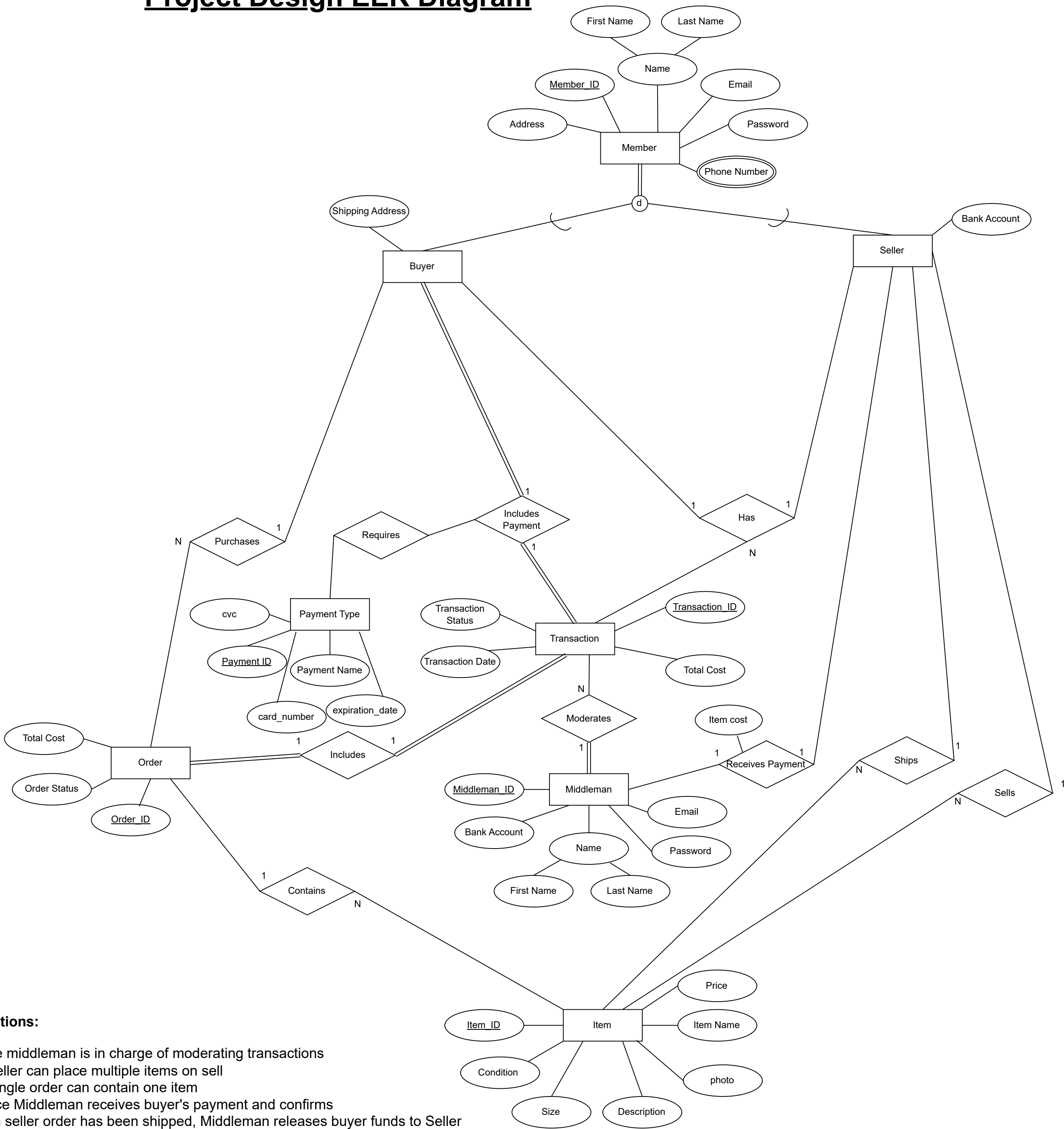
Assumptions:

1. One middleman is in charge of moderating transactions
2. A seller can place multiple items on sell
3. A single order can contain one item
4. Once Middleman receives buyer's payment and confirms
   with seller order has been shipped, Middleman releases buyer funds to Seller
5. Buyer and sellers don't always have to buy and sell
6. All items are unique and never repeated

CHANGES:
- removed category entity to simplify user experience as well as the user inputs these clothing characteristics already.
- removed reviews entity since there is minimal interaction between the buyer and seller, as well as
  the middleman handles that interaction with the funds to ensure trust between buyer and seller.
- added photo attribute to item entity
- added card_number, expiration_date, cvc, created_at attributes to payment type entity
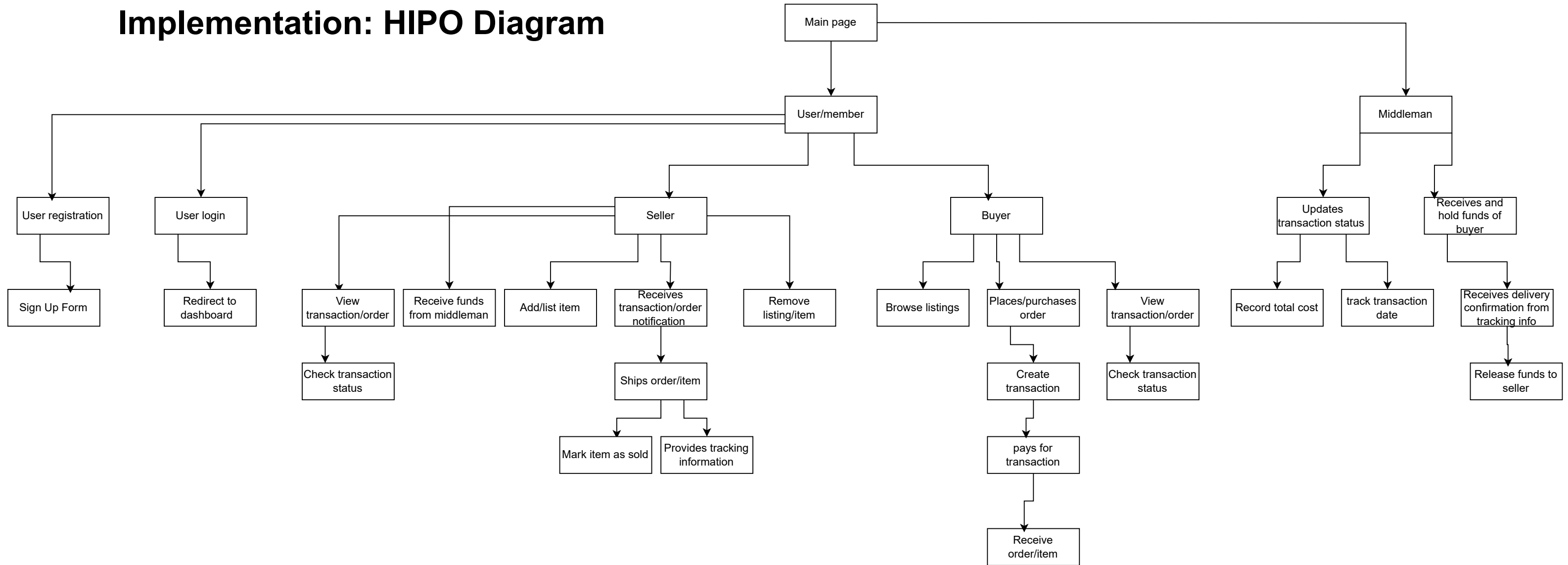
Implementation: RM Diagram

MEMBER

Member_ID	FName	LName	Address	Password	Email

MEMBER_PHONENUMBER

Member_ID	Phone Number

BUYER

Member_ID	Shipping Address

SELLER

Member_ID	Item_ID	Bank Account

TRANSACTION

Transaction_ID	Transaction Status	Transaction Date	Total Cost	Middleman_ID	Order_ID	Member_ID	Payment_ID

MIDDLEMAN

Middleman_ID	FName	LName	Bank Account	Email	Password

PAYMENT TYPE

Payment_ID	Payment name	card_number	expiration_date	cvc	created_at

ORDER

Order_ID	Total Cost	Order Status	Member_ID

ITEM

Item_ID	Item Name	Condition	Size	Description	Price	Order_ID	Sellers_ID	photo

RECIEVES_PAYMENT

Middleman_ID	Member_ID	Item Cost

Assumptions:
- Since mapping 1:1 relationships says "choose one of the relations and include a foreign key in that relation (better to choose an entity with total participation), since there are ties where there are 1:1 relationships with no total participation, the foreign key was included to an arbitrarily chosen entity that participated in that relationship
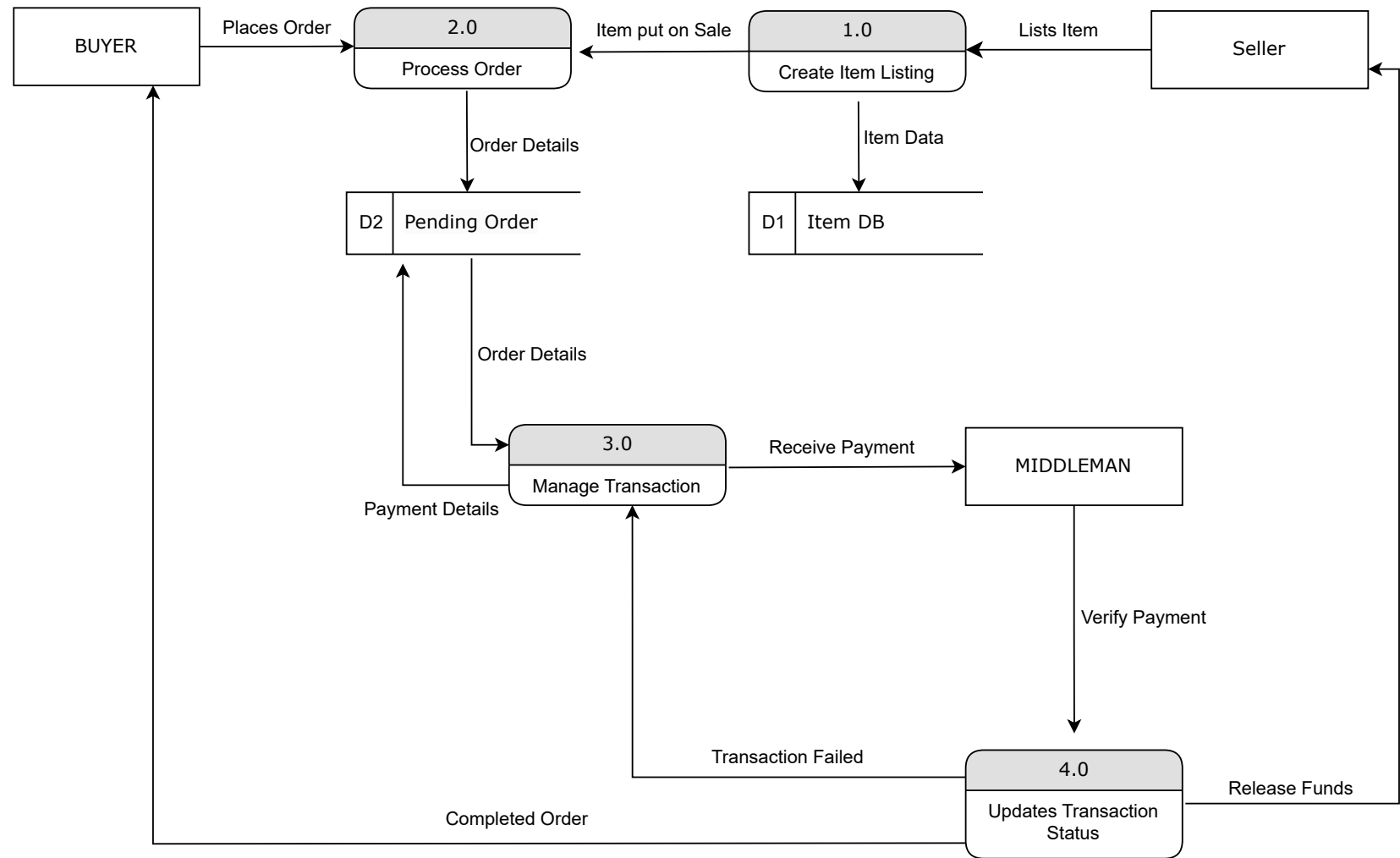- Phone Number multivalued so has its own table

CHANGES:
- In ITEM table, Member_ID was changed to Sellers_ID for clarity
- Removed the category and reviews tables to match with new EER Diagram
- added new photo attribute to item table
- added card_number, expiration_date, cvc, created_at attributes to payment type table

Implementation: HIPO Diagram

```
                                              Main page
                                                  |
                    +-----------------------------+---------------------------------------------+
                    |                                                                           |
                User/member                                                                 Middleman
                    |                                                                           |
   +--------+-------+--------+------------------+                              +----------------+------------------+
   |        |                |                  |                              |                                   |
User      User            Seller             Buyer                      Updates                          Receives and
registration login                                                  transaction status                  hold funds of
                                                                                                             buyer
```

- **Main page** → **User/member** , **Middleman**

- **User/member** → **User registration**, **User login**, **Seller**, **Buyer**

  - **User registration** → **Sign Up Form**
  - **User login** → **Redirect to dashboard**

  - **Seller** → **View transaction/order**, **Receive funds from middleman**, **Add/list item**, **Receives transaction/order notification**, **Remove listing/item**
    - **View transaction/order** → **Check transaction status**
    - **Receives transaction/order notification** → **Ships order/item** → **Mark item as sold**, **Provides tracking information**

  - **Buyer** → **Browse listings**, **Places/purchases order**, **View transaction/order**
    - **Places/purchases order** → **Create transaction** → **pays for transaction** → **Receive order/item**
    - **View transaction/order** → **Check transaction status**

- **Middleman** → **Updates transaction status**, **Receives and hold funds of buyer**
  - **Updates transaction status** → **Record total cost**, **track transaction date**
  - **Receives and hold funds of buyer** → **Receives delivery confirmation from tracking info** → **Release funds to seller**

Implementation: DFD Diagram

Implementation:

For our project, we used MySQL as the DBMS. The SQL statements that we used are below:

```
CREATE TABLE `member` (
  `id` int NOT NULL AUTO_INCREMENT,
  `fname` varchar(50) DEFAULT NULL,
  `lname` varchar(50) DEFAULT NULL,
  `address` text,
  `email` varchar(100) DEFAULT NULL,
  `password` varchar(100) DEFAULT NULL,
  `phone` varchar(15) DEFAULT NULL,
  `role` enum('buyer','seller','middleman') DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
)

CREATE TABLE `items` (
  `item_id` int NOT NULL AUTO_INCREMENT,
  `item_name` varchar(50) DEFAULT NULL,
  `item_condition` varchar(50) DEFAULT NULL,
  `size` varchar(20) DEFAULT NULL,
  `description` varchar(500) DEFAULT NULL,
  `price` decimal(10,2) DEFAULT NULL,
  `item_photo` varchar(255) DEFAULT NULL,
  `member_id` int DEFAULT NULL,
  PRIMARY KEY (`item_id`),
  KEY `member_id` (`member_id`),
  CONSTRAINT `items_ibfk_1` FOREIGN KEY (`member_id`) REFERENCES `member`
(`id`)
)

CREATE TABLE `orders` (
  `order_id` int NOT NULL AUTO_INCREMENT,
  `cost` decimal(10,2) DEFAULT NULL,
  `order_status` varchar(30) DEFAULT NULL,
  `member_id` int DEFAULT NULL,
  `items_id` int DEFAULT NULL,
  `funds_released` tinyint(1) DEFAULT '0',
  `archived` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`order_id`),
```

```sql
  KEY `member_id` (`member_id`),
  KEY `items_id` (`items_id`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`member_id`) REFERENCES `member`
(`id`),
  CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`items_id`) REFERENCES `items`
(`item_id`)
)

CREATE TABLE `payment` (
  `payment_id` int NOT NULL AUTO_INCREMENT,
  `payment_type` enum('credit','debit') NOT NULL,
  `card_number` varchar(255) NOT NULL,
  `expiration_date` varchar(5) NOT NULL,
  `cvc` varchar(255) NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`payment_id`),
  KEY `idx_card_number` (`card_number`)
)

CREATE TABLE `transaction` (
  `transaction_id` int NOT NULL AUTO_INCREMENT,
  `transaction_status` varchar(50) NOT NULL,
  `transaction_date` date DEFAULT NULL,
  `total_cost` decimal(10,2) NOT NULL,
  `order_id` int NOT NULL,
  `member_id` int NOT NULL,
  `payment_id` int DEFAULT NULL,
  PRIMARY KEY (`transaction_id`),
  KEY `order_id` (`order_id`),
  KEY `member_id` (`member_id`),
  KEY `payment_id` (`payment_id`),
  CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `orders`
(`order_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `transaction_ibfk_2` FOREIGN KEY (`member_id`) REFERENCES `member`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `transaction_ibfk_3` FOREIGN KEY (`payment_id`) REFERENCES
`payment` (`payment_id`) ON DELETE SET NULL ON UPDATE CASCADE
)

// Get all members
```

SELECT * FROM member

// Get items based on member where order_status is not 'Completed'
SELECT i.*, o.order_status FROM items AS i LEFT JOIN orders AS o ON i.item_id =
o.items_id WHERE i.member_id = ? AND (o.order_status IS NULL OR o.order_status !=
'Completed')

// Get all items
SELECT i.*, o.order_status FROM items AS i LEFT JOIN orders AS o ON i.item_id =
o.items_id WHERE (o.order_status IS NULL OR o.order_status != 'Completed')

// Create Items
INSERT INTO items (item_name, item_condition, size, description, price, item_photo,
member_id) VALUES (?);

// Sign-up Endpoint
INSERT INTO member (fname, lname, address, email, password, phone, role) VALUES (?, ?, ?,
?, ?, ?, ?)

// Login Endpoint
SELECT * FROM member WHERE email = ? AND password = ?

// Item Delete
DELETE FROM items WHERE item_id = ?

// Items Update
UPDATE items SET item_name = ?, item_condition = ?, size = ?, price = ?, description = ?,
item_photo = ? WHERE item_id = ?

// Verify member exists before creating order
SELECT id FROM member WHERE id = ?

// Verify item exists
SELECT item_id FROM items WHERE item_id = ?

// Create new order
INSERT INTO orders (cost, order_status, member_id, items_id) VALUES (?, ?, ?, ?)

// Get order

SELECT o.order_id, o.cost, o.order_status, o.funds_released, i.item_id, i.item_name, i.price, i.item_photo, b.fname AS buyer_fname, b.lname AS buyer_lname, s.fname AS seller_fname, s.lname AS seller_lname FROM orders AS o JOIN items AS i ON o.items_id = i.item_id JOIN member AS b ON o.member_id = b.id JOIN member AS s ON i.member_id = s.id WHERE o.archived = 0 ORDER BY o.order_id DESC;

// get buyer's orders
SELECT o.order_id, o.cost, o.order_status, i.item_id, i.item_name, i.item_photo, i.price, i.description, i.size, i.item_condition, m.fname AS seller_fname, m.lname AS seller_lname FROM orders AS o JOIN items AS i ON o.items_id = i.item_id JOIN member AS m ON m.id = i.member_id WHERE o.member_id = ? AND o.archived = 0 ORDER BY o.order_id DESC;

// update the order status endpoint
UPDATE orders SET order_status = ?, funds_released = ? WHERE order_id = ?

// set order archive
UPDATE orders SET archived = '1' WHERE order_id = ?

// for seller's orders
SELECT o.*, i.*, m.fname as buyer_fname, m.lname as buyer_lname FROM orders AS o JOIN items AS i ON o.items_id = i.item_id JOIN member AS m ON m.id = o.member_id WHERE i.member_id = ? AND o.archived = 0 ORDER BY o.order_id DESC

// for calculating total profits
SELECT COALESCE(SUM(o.cost), 0) as total FROM orders AS o JOIN items AS i ON o.items_id = i.item_id WHERE i.member_id = ? AND o.order_status = 'Completed' AND o.funds_released = TRUE

// get seller orders
SELECT o.*, i.member_id as seller_id FROM orders AS o JOIN items AS i ON o.items_id = i.item_id WHERE i.member_id = ?

// for seller's orders archived
SELECT o.*, i.*, m.fname as buyer_fname, m.lname as buyer_lname FROM orders AS o JOIN items AS i ON o.items_id = i.item_id JOIN member AS m ON m.id = o.member_id WHERE i.member_id = ? AND o.archived = 1 ORDER BY o.order_id DESC

// Delete all orders related to an item
DELETE FROM orders WHERE items_id = ?

```sql
// delete items
DELETE FROM items WHERE item_id = ?

// Create new payment
INSERT INTO payment (payment_type, card_number, expiration_date, cvc) VALUES (?, ?, ?, ?)

// Create new transaction
INSERT INTO transaction (transaction_status, transaction_date, total_cost, order_id,
member_id, payment_id) VALUES (?, ?, ?, ?, ?, ?)

// fetch buyer transactions
SELECT t.*, p.payment_type, m.fname as seller_fname, m.lname as seller_lname FROM
transaction AS t JOIN orders AS o ON t.order_id = o.order_id JOIN items AS i ON o.items_id =
i.item_id JOIN member AS m ON i.member_id = m.id JOIN payment AS p ON t.payment_id =
p.payment_id WHERE t.member_id = ? ORDER BY t.transaction_date DESC
```

User Manual:

1. You will be presented a landing page which asks you to sign up or login.
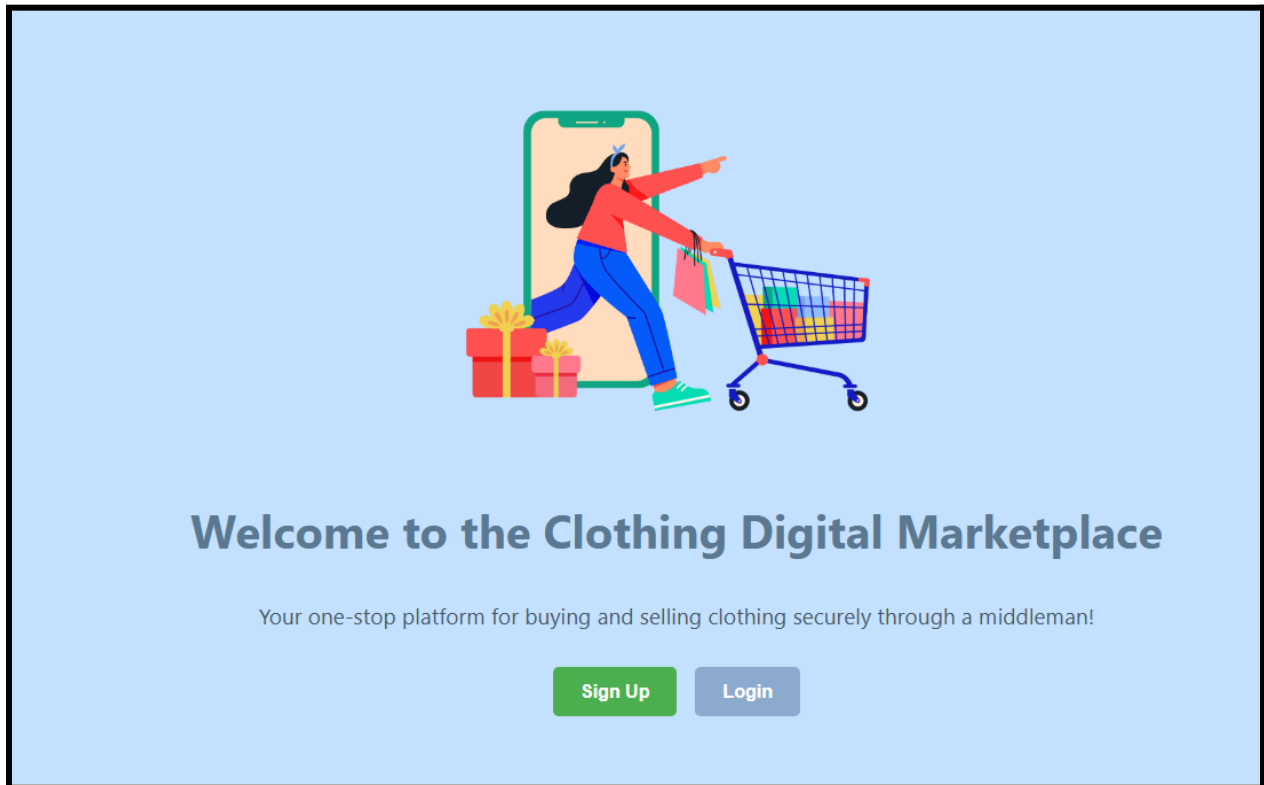
2. Sign Up, where you will be either a seller or buyer.

3. Login, where you will log in based on your account information from the sign up form.

4. After logging in you will be presented with a the buyer or seller or middleman page (if you are the middleman) depending on what you chose during the sign up.

Seller page: On the seller page, sellers will be able to add, modify, and delete items.

Seller page: As well as check orders that buyers have placed, ship orders, and archive orders.
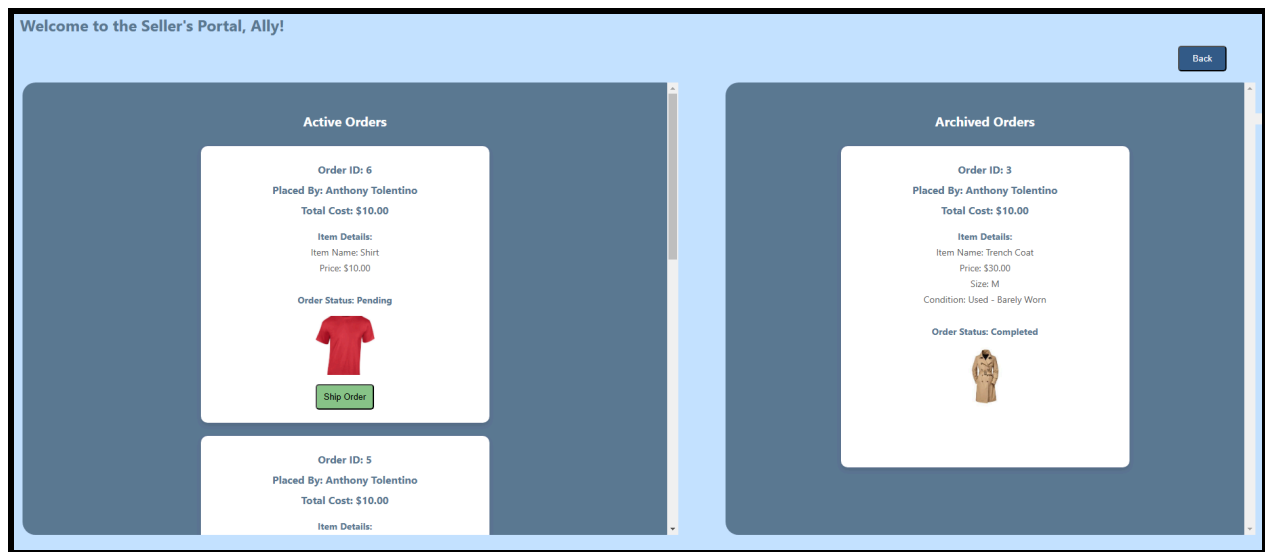
Buyer page: On the buyer page you will be able to browse the different items that have been listed, and add them to your cart.
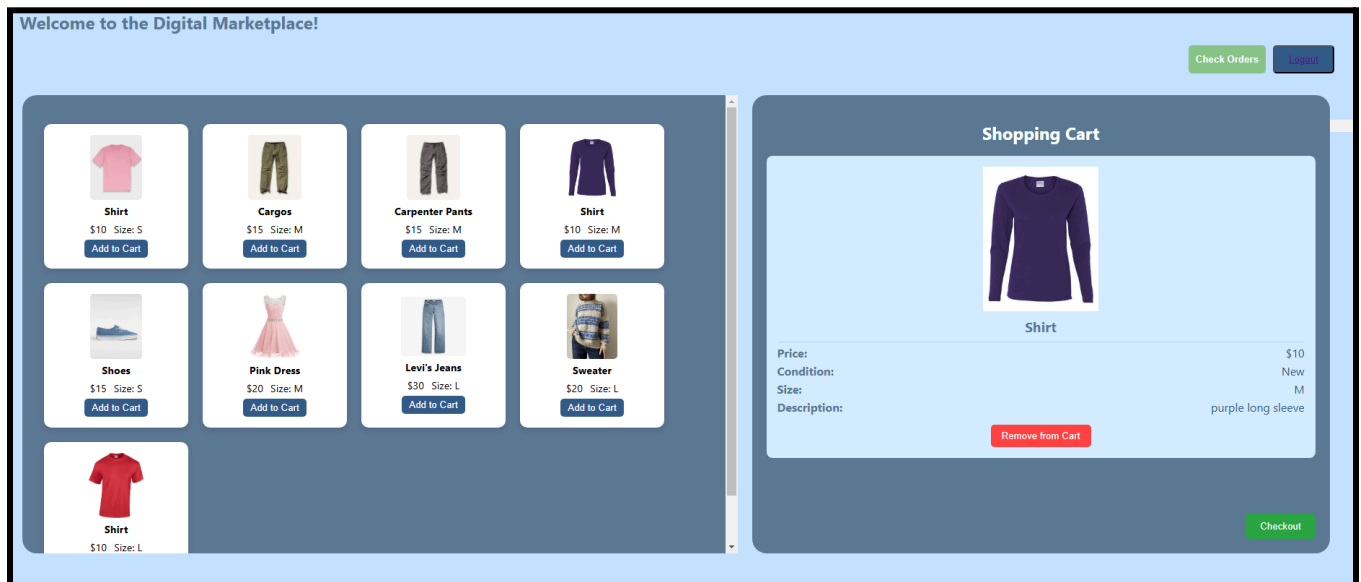
Buyer page: After clicking the checkout button you will see the checkout screen

Checkout

Order Summary

Shirt

Condition:	New
Size:	M
Description: purple long sleeve	
Price:	$10.00

Total Cost: $10.00

Proceed to Payment

Buyer page: After clicking the Proceed to Payment button you will be presented with the payment information screen.

Payment Information

Enter Payment Details

Payment Type

Credit Card

Card Number

1234 5678 9012 3456

Expiration Date

MM/YY

CVC

123

Total Amount: $10.00

Complete Payment

Buyer page: You will also be able to check your current orders and notify the middleman that the item has been received as well as your transaction history.

Middleman page: On the middleman page you will see all the orders and their status as well as the funds status, depending on if the seller has shipped or if the buyer has received the shipped order the middleman can release the funds.

Appendix:

Member Table:

id	fname	lname	address	email	password	phone	role
1	Ally	Pascual	123	arcciepasc@gmail.com	1234	111111111	seller
2	Anthony	Tolentino	123	anthony@gmail.com	1234	111111111	buyer
3	Middle	Man	123	middleman@gmail.com	1234	111111111	middleman
4	Ty	ben	123 Main Street	beenlee@gmail.com	1234	1112223333	seller

Item Table:

item_id	item_name	item_condition	size	description	price	item_photo	member_id
3	Carpenter Pants	Used - Barely Worn	M	baggy grey cargos	15.00	https://truewerk.com/cdn/shop/files/t1_werkpa...	1
4	Shirt	New	M	purple long sleeve	10.00	https://www.budgetpromotion.ca/cdn/shop/pro...	1
5	Shoes	Used - Barely Worn	S	Size US 6 Womens Vans	15.00	https://images.vans.com/is/image/VansEU/VN0...	1
6	Pink Dress	New	M	pink dress perfect for parties	20.00	https://1bb47ed2a0c038093a3e-b3b89d92493...	1
7	Men's shirt	Used - Barely Worn	L	black and white	10.00	https://www.stormtech.ca/cdn/shop/products/...	1
8	Levi's Jeans	New	L	jeans	30.00	https://lsco.scene7.com/is/image/lsco/A556600...	1
9	Sweater	Used - Barely Worn	L	blue and white	20.00	https://www.galthouseofyarn.ca/cdn/shop/files...	1
10	Trench Coat	Used - Barely Worn	M	beige trench coat	30.00	https://encrypted-tbn0.gstatic.com/images?q=...	1
11	Shirt	New	L	red shirt	10.00	https://m.media-amazon.com/images/I/81BWrK...	1

Order Table:

order_id	cost	order_status	member_id	items_id	funds_released	archived
1	15.00	Shipping	2	2	0	0
2	15.00	Order Received	2	5	NULL	0
3	10.00	Completed	2	10	1	1
5	10.00	Completed	2	7	1	0
6	10.00	Pending	2	11	0	0

Transaction Table:

transaction_id	transaction_status	transaction_date	total_cost	order_id	member_id	payment_id
1	Completed	2024-12-08	15.00	1	2	1
2	Completed	2024-12-08	20.00	2	2	2
3	Completed	2024-12-08	10.00	3	2	3
4	Completed	2024-12-08	10.00	4	2	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Payment Table:

payment_id	payment_type	card_number	expiration_date	cvc	created_at
1	debit	1234	02/27	123	2024-12-07 22:09:17
2	debit	111	01/26	112	2024-12-07 22:12:35
3	debit	111	11/27	123	2024-12-07 22:13:10
4	debit	111	01/25	111	2024-12-07 22:25:52