Clothing Store Point of Sale System

Team Members
Drew Miller, Caleb Thai, Nathan Tran, Jason Kao, Parleen Bagga, Adarsh Shresth

System Description

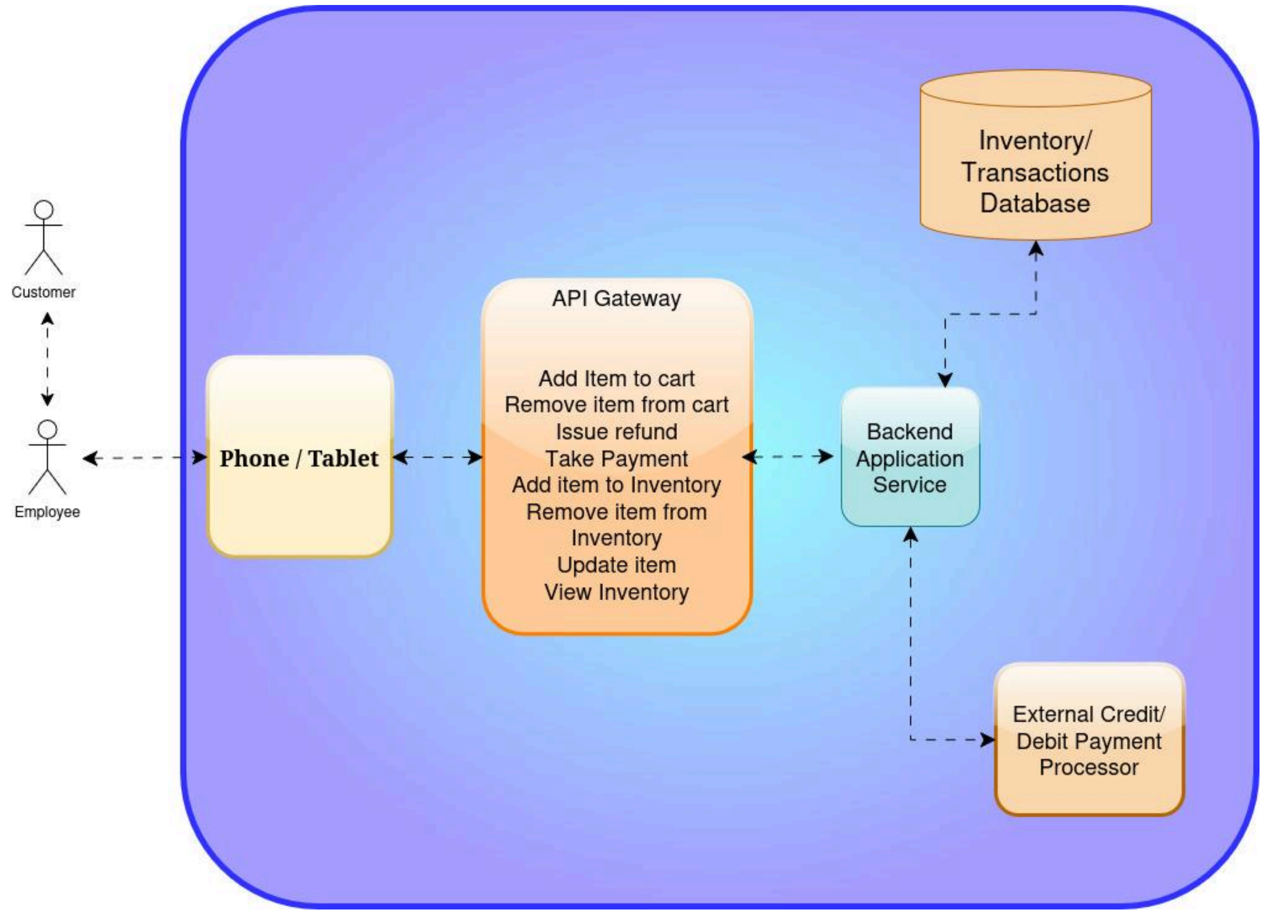
This POS system streamlines sales and inventory management for employees. It supports purchases, returns, and integrates with external processors for credit/debit card payments. Transactions can be completed using cash, card, or barcode scanning/manual entry of item IDs.

Key features include:

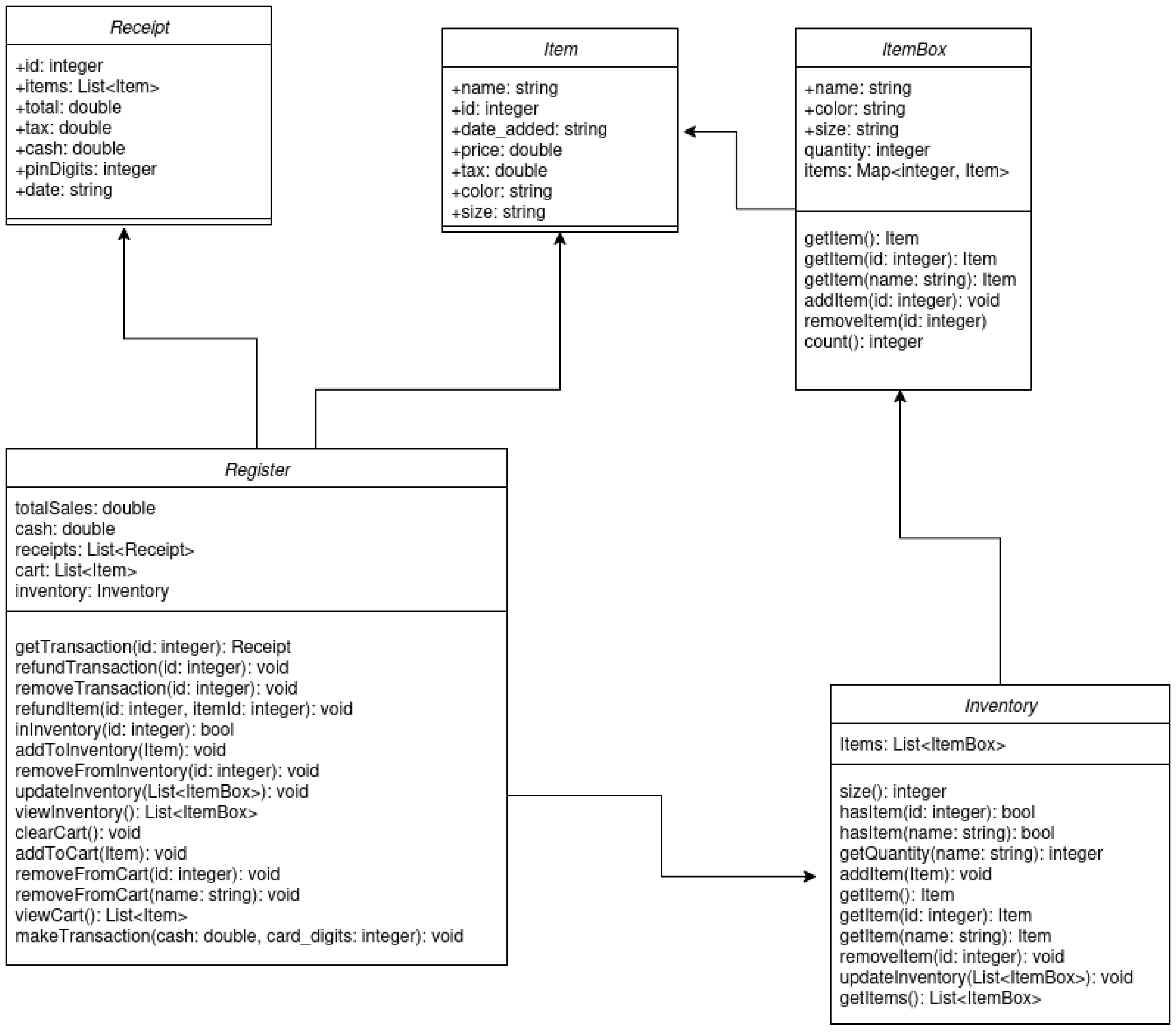
- Automatic calculation of totals with sales tax.
- Real-time inventory updates after sales or refunds.
- Refunds are issued in cash only.
- Employees can search inventory by item ID, name, or date added.
- Staff can add items with details like price, quantity, size, color, and ID.
- Data is stored securely in a cloud-synced database, accessible across store locations.
- Transaction history is securely stored and accessible only to administrators.
- The system works on iOS and Android phones or tablets with internet access and camera barcode scanning.

Software Architecture Overview

- Architectural diagram of all major components



- UML Class Diagram



Receipt
+id: integer
+items: List<Item>
+total: double
+tax: double
+cash: double
+pinDigits: integer
+date: string
Item
+name: string
+id: integer
+date_added: string
+price: double
+tax: double
+color: string
+size: string
ItemBox
+name: string
+color: string
+size: string
quantity: integer
items: Map<integer, Item>
getItem(): Item
getItem(id: integer): Item
getItem(name: string): Item
addItem(id: integer): void
removeItem(id: integer)
count(): integer
Register
totalSales: double
cash: double
receipts: List<Receipt>
cart: List<Item>
inventory: Inventory
getTransaction(id: integer): Receipt
refundTransaction(id: integer): void
removeTransaction(id: integer): void
refundItem(id: integer, itemId: integer): void
inInventory(id: integer): bool
addToInventory(Item): void
removeFromInventory(id: integer): void
updateInventory(List<ItemBox>): void
viewInventory(): List<ItemBox>
clearCart(): void
addToCart(Item): void
removeFromCart(id: integer): void
removeFromCart(name: string): void
viewCart(): List<Item>
makeTransaction(cash: double, card_digits: integer): void
Inventory
Items: List<ItemBox>
size(): integer
hasItem(id: integer): bool
hasItem(name: string): bool
getQuantity(name: string): integer
addItem(Item): void
getItem(): Item
getItem(id: integer): Item
getItem(name: string): Item
removeItem(id: integer): void
updateInventory(List<ItemBox>): void
getItems(): List<ItemBox>

W

Receipt

Description:
Represents a completed transaction. Each receipt contains information about purchased items, payment details, and the date of the sale.

Attributes:

- `id: integer` — Unique identifier for the receipt.
- `items: List<Item>` — Collection of all items purchased in the transaction.
- `total: double` — Total cost including tax.
- `tax: double` — Total tax applied to the transaction.
- `cash: double` — Amount of cash paid by the customer (0 if card only).
- `pinDigits: integer` — Last few digits of the card number used for payment (0 if cash only).
- `date: string` — Date and time when the transaction occurred.

Operations:
(No explicit methods defined for this class; data container for Register operations.)

Item

Description:
Represents a single product in the store with its identifying details and pricing information.

Attributes:

- `name: string` — Name of the item.
- `id: integer` — Unique identifier for the item.

- `date_added: string` — Date when the item was added to the inventory.
- `price: double` — Price of one unit before tax.
- `tax: double` — Tax amount applied to a single unit.
- `color: string` — Color of the item.
- `size: string` — Size of the item.

Operations:
(No explicit methods defined for this class; serves as a data model for inventory and receipts.)

ItemBox

Description:
Stores a group of identical or related items (same name, color, and size). Tracks the quantity of each item variant and allows lookups and modifications.

Attributes:

- `name: string` — Name of the item type.
- `color: string` — Color of the variant.
- `size: string` — Size of the variant.
- `quantity: integer` — Number of items available.
- `items: Map<integer, Item>` — Map of item IDs to their corresponding `Item` objects.

Operations:

- `getItem(): Item` — Returns one available `Item` from the box.
- `getItem(id: integer): Item` — Returns the `Item` with the specified ID.
- `getItem(name: string): Item` — Returns an `Item` matching the given name.
- `addItem(id: integer): void` — Adds an `Item` with the given ID to the box and increases quantity.
- `removeItem(id: integer): void` — Removes the `Item` with the given ID and decreases quantity.
- `count(): integer` — Returns the current number of items in the box.

Inventory

Description:
Represents the store's entire stock of items. Contains multiple `ItemBox` objects and provides methods to search, add, remove, and update inventory data.

Attributes:

- `items: List<ItemBox>` — List of all item boxes currently in stock.

Operations:

- `size(): integer` — Returns the total number of `ItemBox` entries in inventory.
- `hasItem(id: integer): bool` — Checks if an item with the given ID exists in stock.
- `hasItem(name: string): bool` — Checks if an item with the given name exists in stock.

- `getQuantity(name: string): integer` — Returns the quantity of all items with the given name.
- `addItem(item: Item): void` — Adds an item to the appropriate `ItemBox`, creating a new one if needed.
- `getItem(): Item` — Returns an arbitrary or available `Item` from inventory.
- `getItem(id: integer): Item` — Returns the `Item` with the specified ID.
- `getItem(name: string): Item` — Returns an `Item` matching the specified name.
- `removeItem(id: integer): void` — Removes the `Item` with the specified ID from stock.
- `updateInventory(list: List<ItemBox>): void` — Updates or replaces the inventory with the provided list.
- `getItems(): List<ItemBox>` — Returns all `ItemBox` objects currently in inventory.

Register

Description:
Handles all sales operations, including managing the shopping cart, processing payments, handling refunds, and recording receipts.

Attributes:

- `totalSales: double` — Total revenue accumulated from all transactions.
- `cash: double` — Amount of cash currently available in the register.
- `receipts: List<Receipt>` — All completed transaction receipts.

- `cart: List<Item>` — Items currently being purchased in an ongoing transaction.
- `inventory: Inventory` — Reference to the store's inventory used for item operations.

Operations:

- `getTransaction(id: integer): Receipt` — Retrieves a receipt with the given ID.
- `refundTransaction(id: integer): void` — Issues a full refund for a transaction by ID and restores the inventory.
- `removeTransaction(id: integer): void` — Removes a transaction record from the system without processing a refund.
- `refundItem(id: integer): void` — Refunds a specific item by ID and restores it to inventory.
- `addItemToInventory(item: Item): void` — Adds a single item to inventory.
- `updateInventory(list: List<ItemBox>): void` — Updates inventory in bulk with the provided item boxes.
- `viewInventory(): List<ItemBox>` — Returns the current state of inventory.
- `clearCart(): void` — Empties all items currently in the shopping cart.
- `addToCart(item: Item): void` — Adds a specific item to the cart.
- `removeFromCart(id: integer): void` — Removes an item from the cart by ID.

- `removeFromCart(name: string): void` — Removes an item from the cart by name.
- `viewCart(): List<Item>` — Returns all items currently in the shopping cart.
- `makeTransaction(cash: double, card_digits: integer): void` — Completes a sale, generates a new receipt, updates total sales, adjusts cash balance, and clears the cart.

Team Member Tasks/Responsibilities

Drew Miller

- Partitioning members tasks, Architectural Diagram, UML Class Diagram

Caleb Thai

- System Description, Architectural Diagram

Jason Kao

- System Description, Architectural Diagram

Nathan Tran

- System Description, Architectural Diagram

Adarsh Shresth

- UML Class Diagram, Descriptions

Parleen Bagga

- UML Class Diagram, Descriptions