Clothing Store Point-of-Sale System

Software Design Specification

Spring 2024 Group 7

Triet Lieu, Software Engineer

Konrad Kapusta, Software Architect

Prepared for

CS 250: Introduction to Software System

Instructor: Bryan Donyanavard

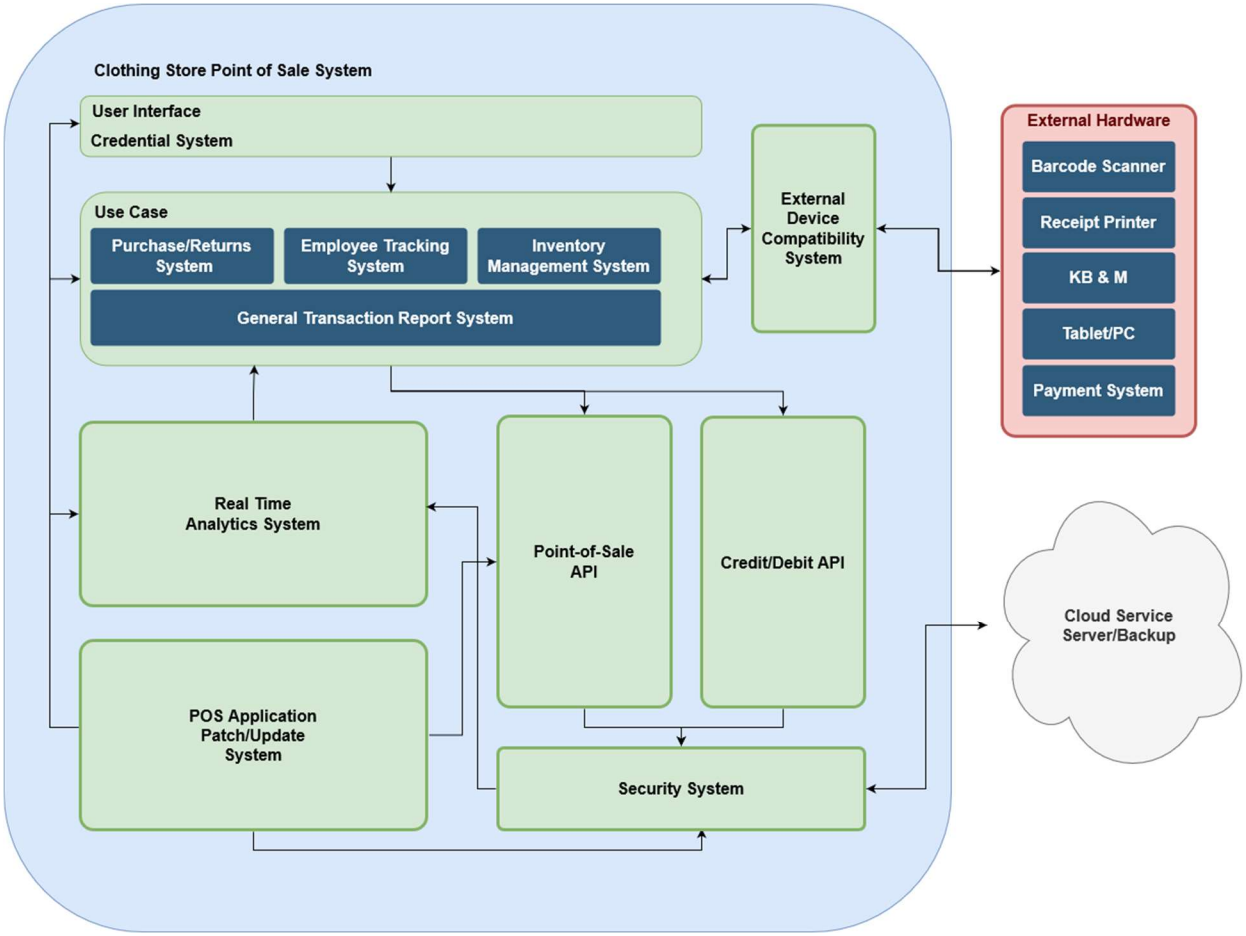
1. System Description

This clothing store point-of-sale system provides a time-saving tool for retail personnel to efficiently conduct all tasks needed to run a physical store, from recording inventory and transactions, to processing purchases, to looking up items. Inventory is automatically updated by integrating with purchases and returns. Items are searchable by the item's ID, price, size, and other attributes a manger sees fit to define.

The system can be installed onto supported barcode scanners and allows for processing card payment through a third-party vendor. An app that supports both iOS and Android can be downloaded into phones and tablets. Both connect to a secure cloud database that backs up and synchronizes data exchange across different store locations.

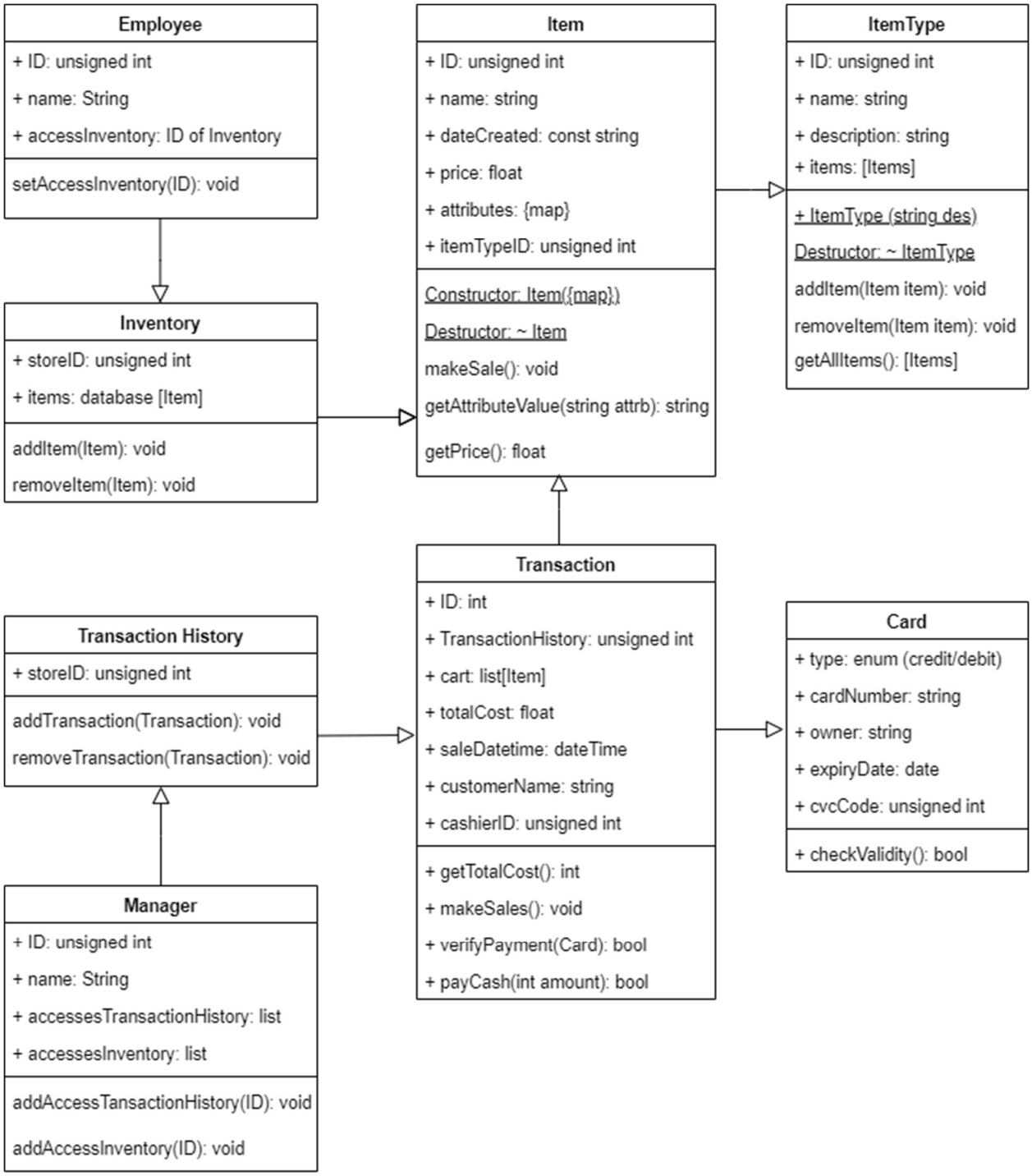
2. Software Architecture Overview

A. Architecture Model



	Description
User Interface	The “view” component of the system with which a user interacts, it should be intuitive and responsive. Receives commands from “controller components”.
Credential System	Manages logins. Use to create or remove users as either Manager or Employee. Must be robust because it grants access to the rest of the system
Purchases/ Returns	Integrates with Inventory and Transaction components and write to their databases automatically.
Employee Tracking	Keeps logs of employee attendance. Ensures actions allowed on the system are only those that relate to authorized store business.
Transaction Report	Give summary accounting of sales, loss, and other metrics that inform the business of which products to stock and how it can improve sales.
Point-of-Sale API	Allows an authorized developer to write software to affect and enhance the functioning of the other components, especially relating to sale and inventory
Credit/Debit Card API	External API developed by third-party vendor that the system calls to verify and charge cards for transaction payments.
External Device	Driver firmware that can be installed into external devices to allow them to integrate with the system. Design requires knowledge of hardware.
Security	Ensures data and the system cannot be hacked.
Cloud Service	Database system to store inventory, transaction history, and other data. Allows the owner of the head of a chain of affiliate business to view the data contributed by individual stores

B. UML Class Diagram



Class	Description
Item	Profile of an item, not the specimen. **attributes** is a map whose keys are attributes like 'size', 'color', and values are specs like 'LG', 'red' for Item. **getPrice()** returns price float. **ID** can be assigned by a database. **dateCreated** is a constant string set as profile's creation date. **itemTypeID** links Item to ItemType via ItemType's unsigned int ID **Constructor** receives input of map {attribute: spec, etc} to set attributes **delete Item** deletes Item, but must be separately removed from Inventory **makeSale()** decreases the count of the Item in inventory by 1 **getAttributeValue(attrb):** eg getAttributeValue('size') → 'small'
ItemType	eg "women shoes", "men dress pants". Use **getAllItems()** to field a list of related Items to suggest to customers. **description** is a string that explains this ItemType in greater detail **items** stores a list of Item profiles belonging to this ItemType **Constructor** takes string input and assigns it to description member. **Destructor** calls removeItem(..) on all Item references in items member list. **(add/remove)Item(Item)** puts input Item into or removes it from ItemType **getAllItems()** returns Items under this ItemType
Card	Denotes a card that is used to pay for a transaction. **type** is an enum that identifies card as of credit or debit type **owner**: string, **expiryDate**: date, **cvcCode**: unsigned int, and **cardNumber:** string are credential details linked to the card

	checkValidity() returns True if all the credential details pass verification.
Transaction	Denotes a register checkout. Items comprise a Transaction, a Card can be used as the payment method, and an Employee serves as the cashier. **cart** stores the list of item purchases. **saleDatetime** tracks the purchase time. customerName is a string. **cashierID** is unsigned int that links to an Employee. **getTotalCost()** returns the sum of all Item's prices. **makeSales()** calls makeSale() of all Items in cart to reduce their inventory count **verifyPayment(Card)** has input Card call checkValidity() to verify card **payCash(int amount)** returns True if the cash amount can cover the total cost
Transaction History	Stores Transactions for Store referenced by **storeID** unsigned int **[add/remove]Transaction(..)** [adds/removes] reference to that Transaction
Inventory	Tracks Items in **items** database for Store referenced by **storeID** unsigned int **[add/remove]Item (Item)** [adds/removes] reference to that Item
Manager	An admin user who can access multiple TransactionHistory and Inventory **accessesInventory**: list of ID of Inventory to which this Manager has access **accessesTransactionHistory**: list of ID of TransactionHistory granted access **addAccess[TransactionHistory/Inventory]** (ID): Grant this Manager access to [TransactionHistory/Inventory] referenced by input ID
Employee	A non-admin user with access to only 1 Inventory referenced by **permission** **setPermission(ID)**: Set Inventory that this Employee can access

3. Development Timeline