Clothing Store Point of Sale System

Prepared by: Alan Shami

March 10, 2023

System Description

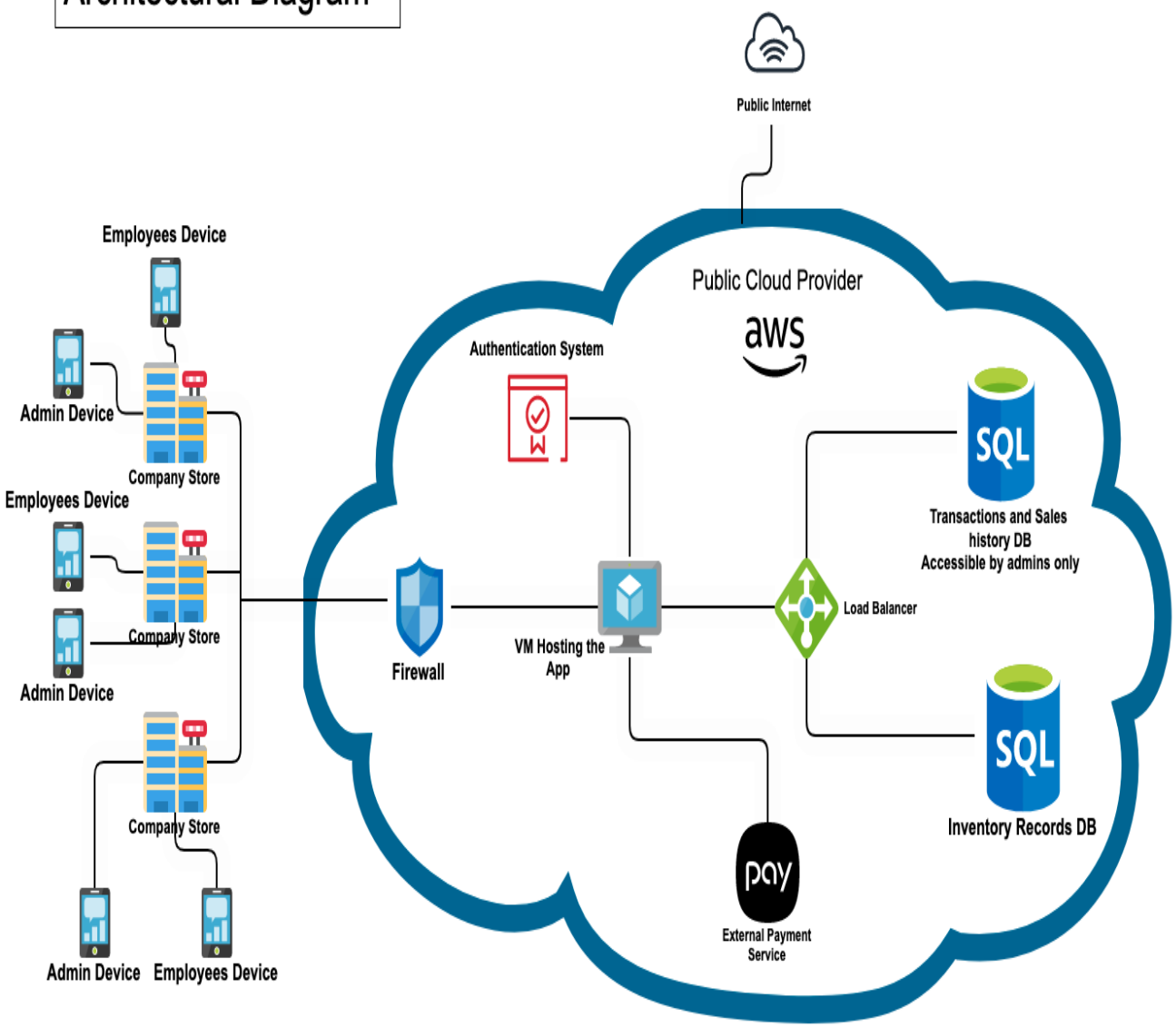
This mobile app works across multi-platforms (iOS and Android) and intended for internal use only by the company employees. The app is designed for a clothing store with multiple locations across the city. The app has two levels of authorization. A basic auth for non-managerial employees, and an admin auth for managerial employees. The basic auth will allow employees to checkout customers, issue refunds, taking into consideration sales tax in both cases, by scanning the barcode of an item of a product using the device camera as a scanner. Also, the app can issue refunds for items from different locations. The app can also accept different payment methods. The payment will be processed through an external payment processor service. However, refunds will be cash only. After each transaction (whether a purchase or a sale transaction), the inventory of the store should be updated. The inventory data of all the store chains will be stored and backed up on a public cloud provider using a relational database. This will enable the app to allow employees to look up an item by its ID, item name, or date added to the inventory. Another relational database will record transaction history and sales number for each store, and it will be accessible by administrator employees only.

Software Architecture Overview

Architectural Diagram, Major Components:

The system will be hosted on a public cloud provider service, for ease maintenance and reduce overhead. This will allow every connection to consciously connect to the same datacenter. The app will be hosted on a virtual machine that connects to firewall, a load-balancer, an external payment processor service, and an auth system. The firewall will allow only traffic from the store network for security purposes. Also, there will be two database instances. The first one will be for transactions history and sales records, and the second one will be for inventory across all store locations. Both databases will be connected to the App host VM through a load-balancer to distribute the traffic based on the authentication of the user. The database will synchronize the inventory and sales transactions with all locations in real-time.

Architectural Diagram



UML Class Diagram:

The main classes for the system include Item, Inventory, Transaction History, Employee, Admin, and Camera.

Item:

This class holds all the variables relating to aspects of the product being sold in the store including name, ID number, price, size, and color. It also holds methods to return those individual values relating to the Item. This class is used by the Inventory, Employee, and Transaction History classes.

Inventory:

Inventory adds or removes items from the inventory of products and has the attributes “date added” and “quantity”. It uses the Item class and is updated by the Employee class.

Transaction History:

Transaction History has operations to update the transaction history and allows the administrator to search for a particular transaction using the sale number or date. Transaction History is only accessible by the Admin class

Employee:

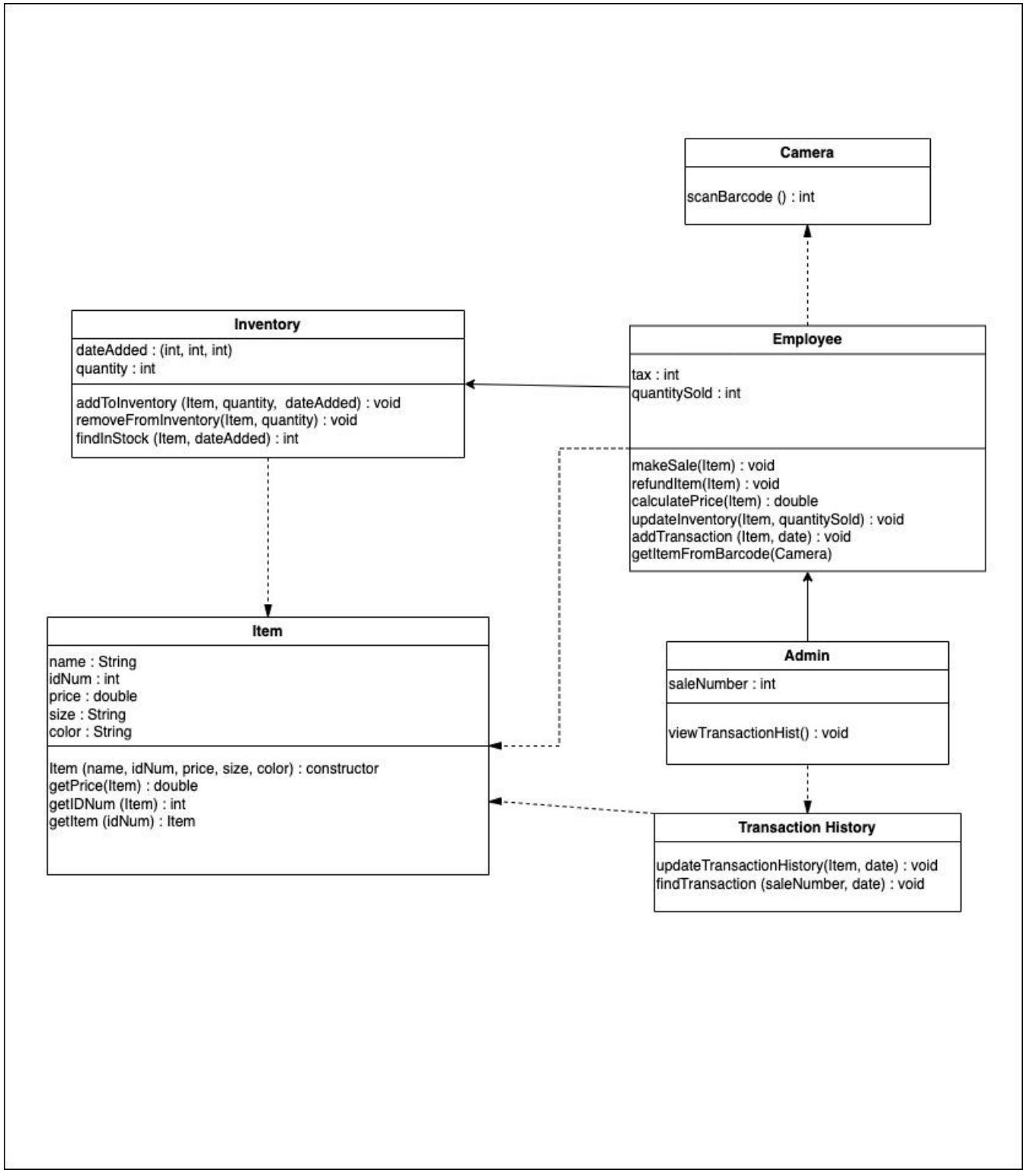
The attributes of the Employee class include tax and quantity sold. The operations of the Employee class cover the aspects of making a sale, updating the inventory and transaction history, and making refunds at the store. It utilizes the Item class and the Camera class, and it regularly updates the Inventory.

Admin:

The Admin class is the only class with the ability to view the transaction history. It contains the attribute of the sale numbers to find a transaction.

Camera:

The Camera class connects the system to the camera of the device which it uses to scan barcodes on items to retrieve the ID number of the item. It is used by the Employee class when making transactions.


Camera
scanBarcode () : int
Inventory
dateAdded : (int, int, int)
quantity : int
addToInventory (Item, quantity, dateAdded) : void
removeFromInventory(Item, quantity) : void
findInStock (Item, dateAdded) : int
Employee
tax : int
quantitySold : int
makeSale(Item) : void
refundItem(Item) : void
calculatePrice(Item) : double
updateInventory(Item, quantitySold) : void
addTransaction (Item, date) : void
getItemFromBarcode(Camera)
Item
name : String
idNum : int
price : double
size : String
color : String
Item (name, idNum, price, size, color) : constructor
getPrice(Item) : double
getIDNum (Item) : int
getItem (idNum) : Item
Admin
saleNumber : int
viewTransactionHist() : void
Transaction History
updateTransactionHistory(Item, date) : void
findTransaction (saleNumber, date) : void

Development Plan & Timeline

The timeline for the project is dependent on the client. This development plan outlines the individual tasks for a three person team to effectively complete the system.