ShopEase

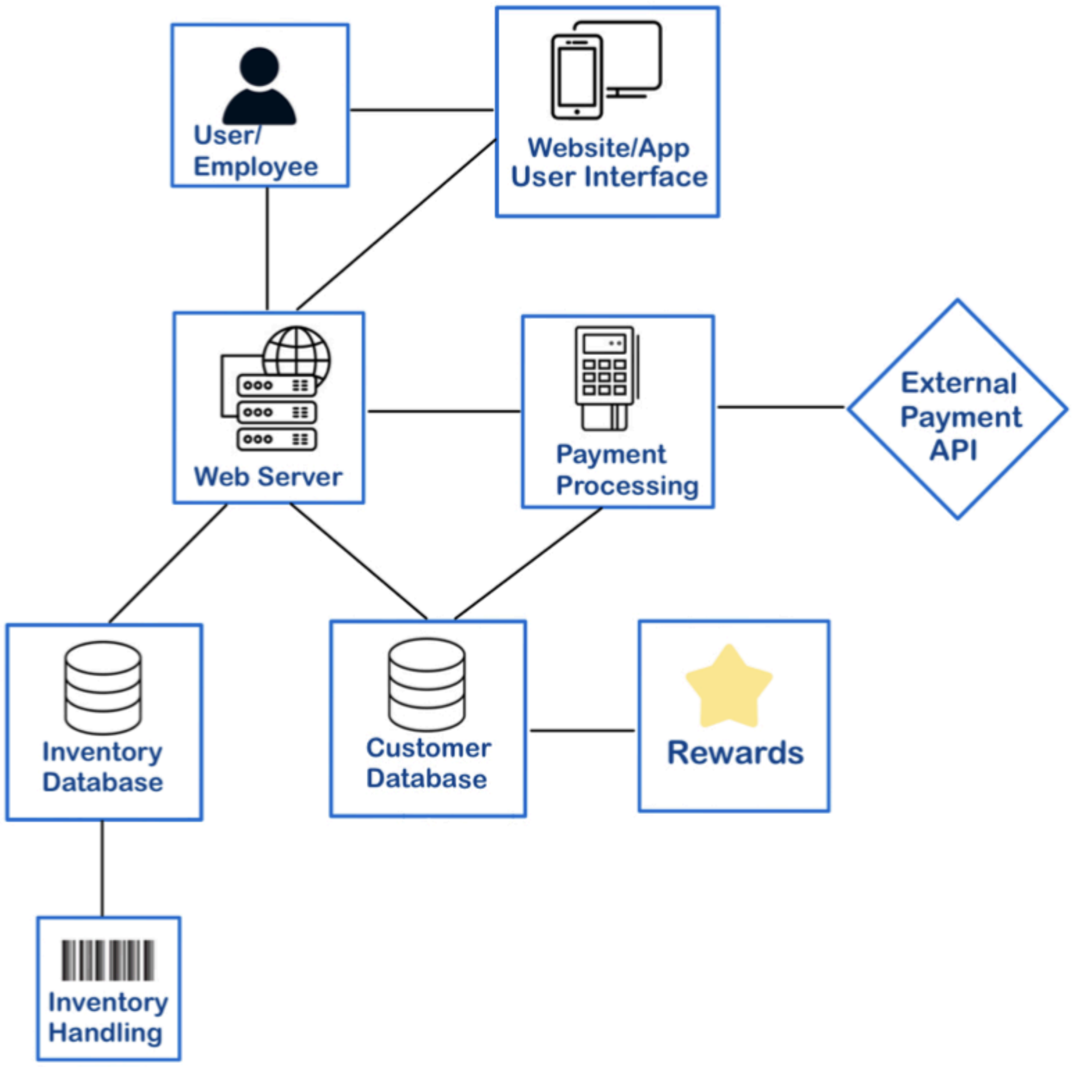
Phoebe Griffiths & Han Luu

System Description

ShopEase is a cutting-edge software system that enhances the buying and selling experience for both the consumer and the seller. Designed for tablets and smartphones, this system pushes a simple user interface for seamless interactions. This system includes key features such as: managing inventory, recording history of customer purchases, supporting secure payment methods, tracking company sales and communicating sales reports to administrators, and a rewards system.

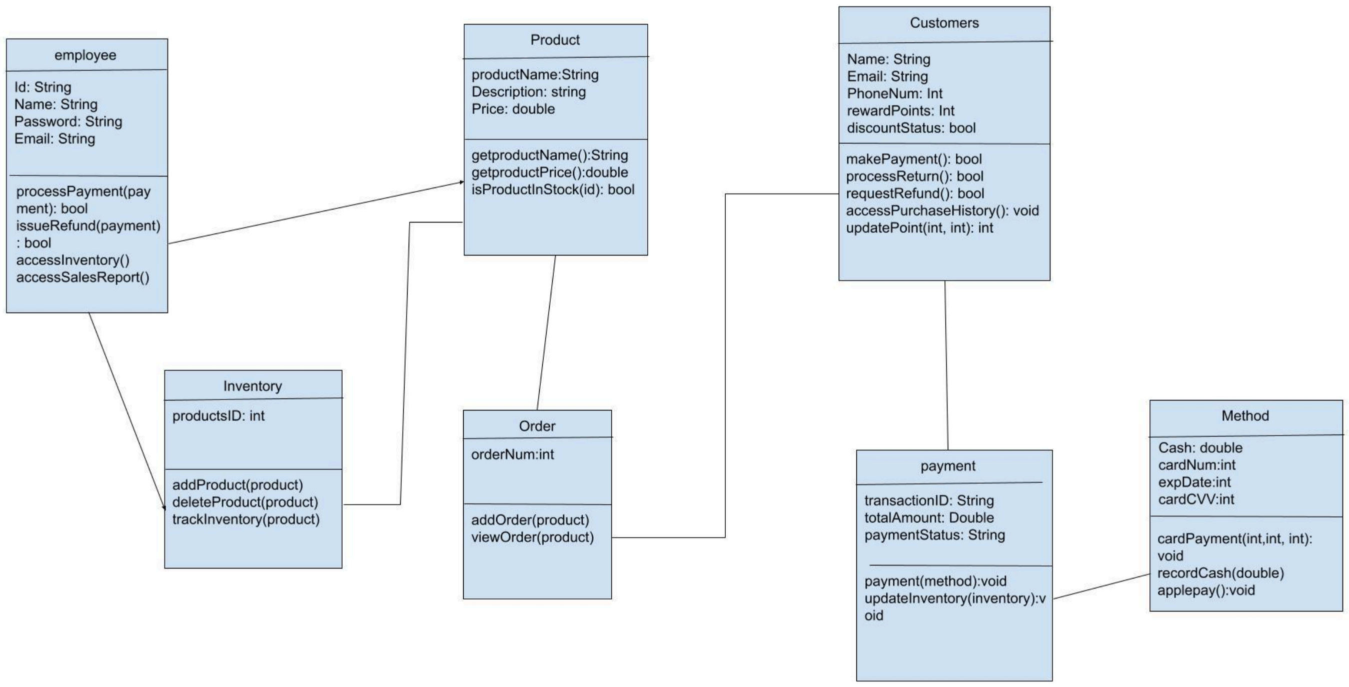
Software Architecture Overview

Software Architecture Diagram:



SWA Description:
The software architecture diagram visually represents how all components of the software system interact and connect with one another. To break down the architecture of ShopEase, we can start with the user/employee module. Both customers and employees interact with the system through the website or app, so this connection represents the facilitation of user interaction. The website/app module provides a user interface accessible to Android and Apple users. This module additionally connects to the web server module to allow for data exchange between the server and user interface. The web server acts as a connector between front-end and back-end development. Web server connects to multiple modules for seamless information flow. The payment processing module supports secure transactions via credit card, but also connects to an API for other payment methods such as Apple pay. As for databases, the inventory module keeps track of product availability and connects to the web server to update inventory after transactions are made. The customer database stores customer information including purchase history, and connects to the rewards modules to retrieve customer reward points.

UML Class Diagram:



1.Employee class: This class covers most interactions made between the employee and the software system. It holds essential attributes regarding employee identification. Its key functionalities allow employees to perform tasks related to transactions, inventory management, and sales analysis.

a. Attributes:
    i. Id: String type, representing number identification of employee
    ii. Name: type of string representing the name of the employee.
    iii. Password: String type, indicating employee password to enter the system
    iv. Email: type of string representing the email address of the employee
b. Operations:
    i. processPayment(payment): initiates the payment process for the employee. Has return type of boolean which indicates if the payment was successfully made
    ii. issueRefund(payment): return type boolean; state whether or not a refund is issued to the customer.
    iii. accessInventory(): this allows for employees to interact with and retrieve information from the product inventory. It would grant authorized personnel to access details about products.
    iv. accessSalesReport(): this operation allows for employees to view details on sales reports.

2.Inventory class: The inventory class is one that is often modified as it relates to live availability of product. It works closely with the product class, which is often a parameter in its operations. This class emphasizes the management of products when they are sold or even missing, always making sure accurate numbers are reflected to physical quantities of items.

a. Attributes:
    i. productsID: int type, product Id number
b. Operations:
    i. addProduct(product): responsible for adding a new product to the inventory. It takes in the product as a parameter, representing the details of the product to be added.
    ii. deleteProduct(product): responsible for removing a product from the inventory.
    iii. trackInventory(product): is designed to monitor and manage product inventory.

3. Customers Class: facilitates various interactions between customers and the shopping store software system. It manages customer information, handles payment transactions, facilitates refunds and returns, provides access to purchase history, and manages reward points accumulation and redemption.

a. Attributes:
    i. Name: type of string representing the name of the customer.

   ii. Email: type of string representing the email address of the customer.
   iii. PhoneNum: has type of int representing the phone number of the customer.
   iv. rewardPoints: has type of int representing the reward points accumulated by the customer.
   v. discountStatus: has boolean type indicating whether the customer is eligible for a discount.

b. Operations:
   i. makePayment(): initiates the payment process for the customer. Has return type of boolean which indicate if the payment is successful made
   ii. processReturn(): handles the return process for the customer. Return type of boolean, indicating whether the return process is successful
   iii. requestRefund(): requests a refund for the customer. Return type of boolean, indicate if the refund is successful requested or not
   iv. accessPurchaseHistory(): allows the customer to access their purchase history.
   v. updatePoint(int, int): updates the customer's reward points based on a specified action.

4. Payment Class: include of the information about the transactions such as total amount, payment status, and the id

a. Attributes:
   i. transactionID: has string type representing the unique id number for the payment transaction.
   ii. totalAmount: has double type representing the total amount of the payment.
   iii. paymentStatus: A string attribute representing the status of the payment transaction.

b. Operations:
   i. payment(method): processes the payment using a specified payment method.
   ii. updateInventory(inventory): updates the inventory after a successful payment transaction.

5. Method Class: process all payment types such as cash, card, or apple pay

a. Attributes:
   i. Cash: has double type, representing the amount of cash receive
   ii. cardNum: has int type, representing the card payment’s number
   iii. expDate: has type int, representing the card expirations date number
   iv. cardCVV: typt int, representing card security code

b. Operations:
    i. cardPayment(int,int,int): have parameter of int, int, int and is used to handle card payment
    ii. recordCash(double): have parameter of double is used to record cash payment from customers
    iii. applepay(): handle payment from Apple Pay transactions

6. Product class:

a. Attributes:
    i. productName: string type, representing name of the product
    ii. Description: string type, representing product description
    iii. Price: double type, representing product's price
b. Operations:
    i. getproductName(): return type of String, the method retrieve the product name
    ii. getproductPrice(): return type is double, the method retrieve the price of the product
    iii. isProductInStock(id): parameter is id- the id of the product to check, return type is boolean, this method check if the product is still in stock before sell to the customer

7. Order class:

a. Attributes:
    i. orderNum: int type, representing order id
b. Operations:
    i. addOrder(product): parameters product-the product to add to order. This method add product to order
    ii. viewOrder(product): method that display the added product in the order

Development plan and timeline

As partners for the ShopEase software system, we took a collaborative approach to this assignment. Each of us independently crafted our own diagram, and then merged the two drafts together to leverage the best of both perspectives. While most tasks were done together, the Software Architecture (SWA) diagram was mainly produced by Phoebe while Han is responsible for the UML diagram. Additionally, Phoebe oversaw document neatness, while Han kept the group on schedule to finish our work in a timely manner. Diagram descriptions were a joint effort.