Software Design Specification
For
ThriftWerks: Point-of-Sale System

Team members:

Ricardo Lemus

Jane Ho

David Kaauwai

System Description

This software, which will be known as "ThriftWerks", is a point-of-sale system that will facilitate the work of retail employees, most notably with processing sales, refunds, and inventory searching. The system can be accessed at front desk registers or through phones and tablets. From there, a login screen will appear where an employee ID and password must be inputted to proceed. The system will not only determine whether the account information matches what exists in the Employee Information Database, but will check if the ID matches that of a higher-level employee, such as a manager, so that more functions inaccessible by regular employees may be unlocked. Once the inputted information is confirmed to be valid, the system will direct the user to the main menu, which displays options of "Sales", "Item and Money Returns", and "Total Stock of Item", all of which will utilize the Inventory Database. Higher-level employees will also have the options of "Management" and "System Maintenance". "Sales" will use a payment API with the Inventory DB to carry out customer transactions when they purchase any item(s), and "Item and Money Returns" will directly access said DB to process customer refunds. "Total Stock of Item" will help search for specific items in the Inventory DB.

Software Architecture Overview

Architectural diagram of all major components:
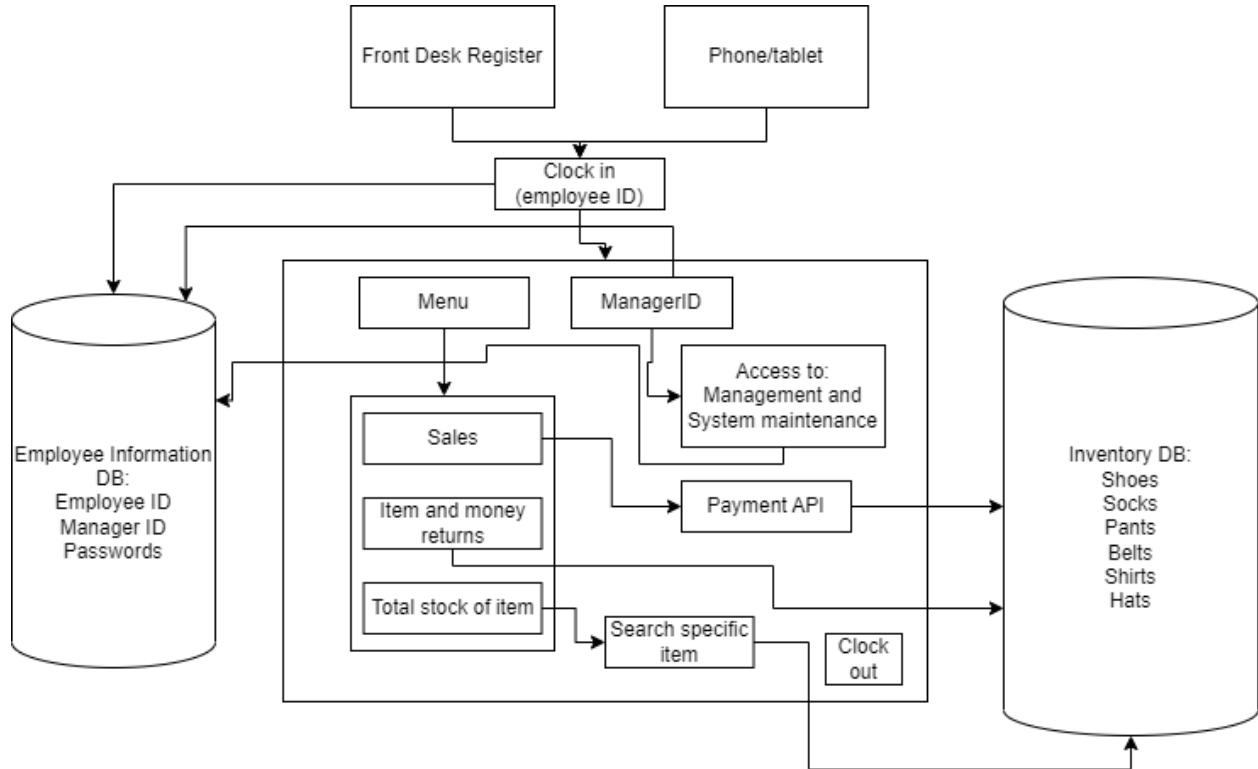
The above software architecture diagram shows all the major components of the system we are designing. It shows what happens when a certain user logs in (manager or employee), two databases that store different information and specifically what information, and the different actions an employee or manager can do such as giving refunds or checking inventory on an item. Below are different characteristics that go more in depth on what each component does in the software architecture diagram.

Description of the software architecture diagram:

Front desk register: Connects to the "Clock in" screen which is to verify that the user is an employee of the store.

Phone/tablet: Also connects to the "Clock in" screen to verify they are an employee.

Clock in (Employee ID): Extends to the "Employee Information DB" to see if the ID and password match an account. Once logged in the "Menu" is accessible.

Employee Information DB: Stores employees/managers credentials.

Manager ID: Extends to the "Employee Information DB" to verify they are a manager and see if an ID and password match. Manager ID also extends to the special tools managers have.

Access to: Management and system maintenance: Allows managers to access moderations, employee management and system maintenance.

Inventory DB: Database where all stock of clothing is kept at.

Menu: Extends to sales, item and money returns (refunds) and total stock of an item (inventory).

Sales: Goes into "Payment API" which is used to process purchase transactions and it also extends into the "Inventory DB" to update the stock on an item.

Payment API: Used to process transactions.

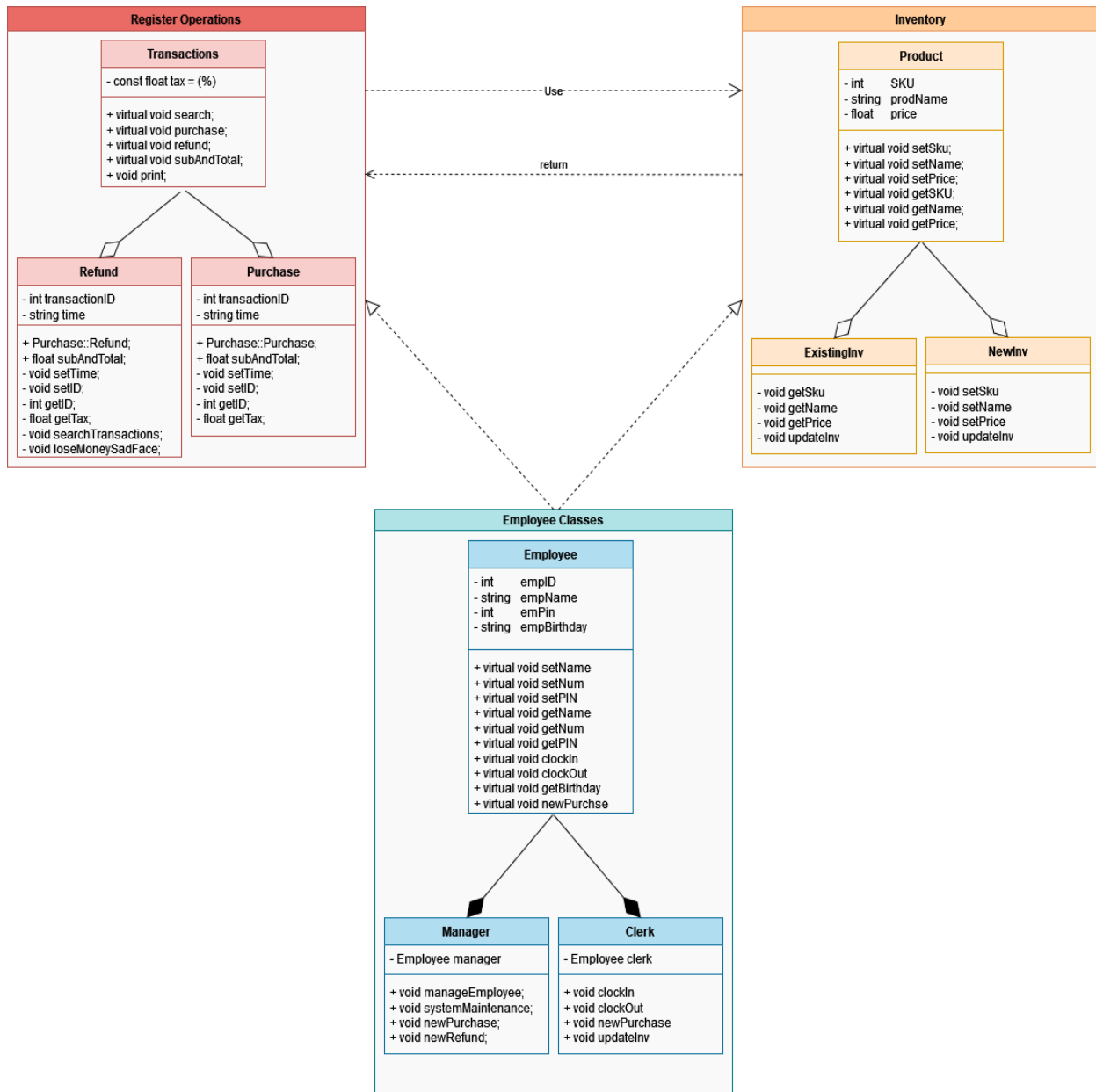**Item and money returns:** Extend into "Inventory DB" to update the stock when an item is returned.

Total stock of an item: Goes into "Search specific item" which searches in the database of "Inventory DB" for a particular item.

Search specific item: Searches inside "Inventory DB" for an item the user wants to find.

Clock out: Logs a user out to prevent any non employee from tampering with the system.

UML Class Diagram

Register Operations

Transactions
- const float tax = (%)

+ virtual void search;
+ virtual void purchase;
+ virtual void refund;
+ virtual void subAndTotal;
+ void print;

Refund
- int transactionID
- string time

+ Purchase::Refund;
+ float subAndTotal;
- void setTime;
- void setID;
- int getID;
- float getTax;
- void searchTransactions;
- void loseMoneySadFace;

Purchase
- int transactionID
- string time

+ Purchase::Purchase;
+ float subAndTotal;
- void setTime;
- void setID;
- int getID;
- float getTax;

Use →

← return

Inventory

Product
- int   SKU
- string   prodName
- float   price

+ virtual void setSku;
+ virtual void setName;
+ virtual void setPrice;
+ virtual void getSKU;
+ virtual void getName;
+ virtual void getPrice;

ExistingInv
- void getSku
- void getName
- void getPrice
- void updateInv

NewInv
- void setSku
- void setName
- void setPrice
- void updateInv

Employee Classes

Employee
- int   empID
- string   empName
- int   emPin
- string   empBirthday

+ virtual void setName
+ virtual void setNum
+ virtual void setPIN
+ virtual void getName
+ virtual void getNum
+ virtual void getPIN
+ virtual void clockIn
+ virtual void clockOut
+ virtual void getBirthday
+ virtual void newPurchse

Manager
- Employee manager

+ void manageEmployee;
+ void systemMaintenance;
+ void newPurchase;
+ void newRefund;

Clerk
- Employee clerk

+ void clockIn
+ void clockOut
+ void newPurchase
+ void updateInv

Description of classes:

Transactions: Base class for common transactions made by employees, which includes processing purchases and refunds, searching inventory, and calculating purchase/refund totals.

Refund: Child class of Transactions, includes functions for searching refunds made, getting/setting IDs of refunds, and setting times for them.

Purchase: Child class of Transactions, includes functions for getting/setting IDs for purchases made and setting times for them.

Product: Base class for basic product information that is in the inventory, such as SKU codes, product names, and prices.

ExistingInv: Retrieves information of an existing inventory item, important for searches.

NewInv: Sets information for a new inventory item.

Employee: Base class for employee account information, which includes ID numbers, names, PINs, and birthdays.

Manager: Child class of Employee, gives access to manager-only functions like system maintenance and managing employee information.

Clerk: Child class of Employee, includes functions for clocking in and out for work.

Description of attributes:

const float tax (Transactions): Will be accessed to calculate purchase/refund totals with tax.

int transactionID (Refund/Purchase): The ID assigned to a specific refund/purchase.

string time (Refund/Purchase): The time at which the refund/purchase was made.

int SKU (Product): The product's identifying code, will mainly be used for inventory searches.

string prodName (Product): The name of a product, can also be used for searching.

float price (Product): The price of a product.

int empID (Employee): The ID number of an employee, will be needed for login.

string empName (Employee): The name of an employee.

int empPIN (Employee): The PIN/password of an employee, will be needed for login.

string empBirthday (Employee): The birth date of an employee.

Employee manager (Manager): An object of an employee, specifically for managers. Should have access to manager-specific functions.

Employee clerk (Clerk): An object of an employee, specifically for clerks.

Description of operations/connectors:

Development plan and timeline

Partitioning of tasks:
We will first start with having the task of making the UML class diagram and the Software Architecture diagram. Then for our final task, we will write the descriptions.

The goal is to have the diagrams started when we first met for group work and finished by Wednesday (3/7/2023) and spend the next couple days adding descriptions. Then on Friday, have everything ready to turn in before midnight.

Team member responsibilities:

This project was split evenly between the three of us.

Ricardo Lemus:
- Architectural diagram of all major components
- Description of the software architecture diagram

Jane Ho:
- Brief overview of system
- Description of attributes

David Kaauwai:
- UML Class Diagram
- Description of classes
- Description of operations