Point of Sale System for General Restaurant

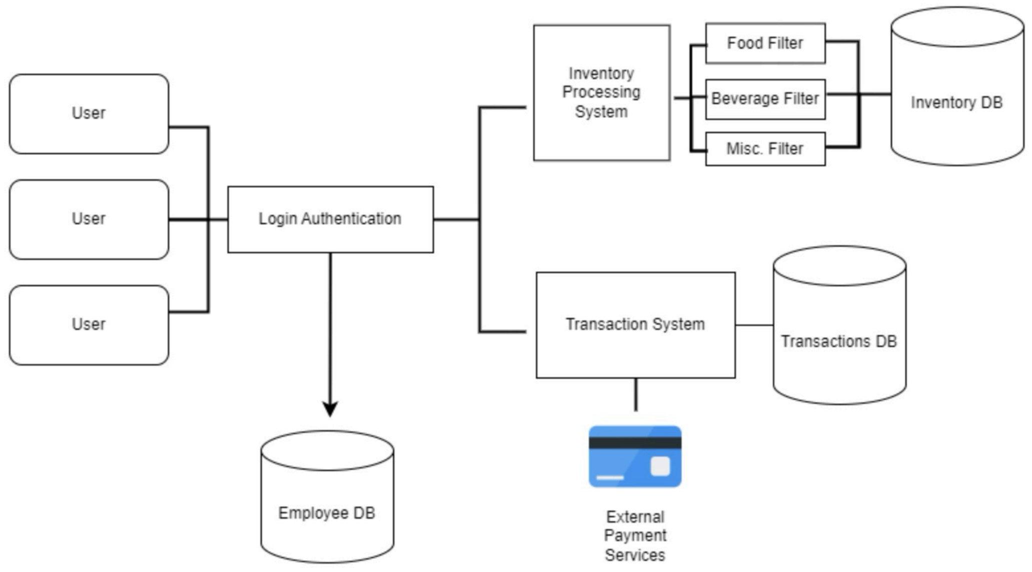
Team Members: Husain Patanwala, Andrew-Jacob Santos, Cody Tran

System Description

This system is being developed and designed to be able to support general restaurant management functions. It will be able to take input from users, and then the system will interact with the backend updating and fetching information from databases to support daily restaurant operations. Ensuring smoother workflow between front-of-the-house and back-of-the-house employees. The users will access this system through tablet stations throughout the restaurant or through portable tablets.

Software Architecture Overview

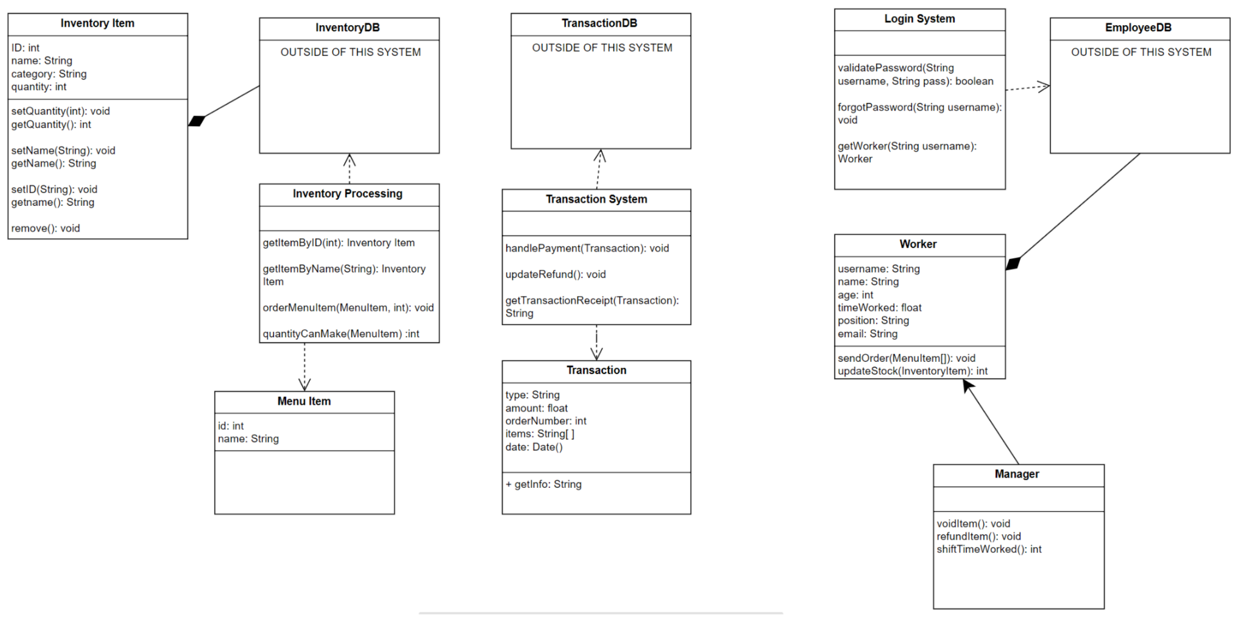
Architectural Diagram of Major Components:



The leftmost component in the diagram represents the users (in this case restaurant staff) who will be provided input to the system through various event triggers. Initially, users will have to enter the PoS through the login authentication system which depends on the Employee database. This db houses login info for each staff member, paycheck, position, and other relevant information. The other components of the system can only be accessed after passing through this step. Afterward, users can either follow one of two paths: accessing inventory or using the transaction system.

The IPT (inventory processing system) includes the functions necessary to interact with system data.The IPT is also associated with some general filters that interact with the inventory database. These filters limit the scope of what the user might be searching for, before entering the database itself. On the other hand, the transaction system will have necessary functionality to interact with third party payment systems. However, it depends on the info stored in the database to issue refunds and such.

UML Class Diagram:



Inventory Item:

The class that represents a particular raw inventory item. These will include raw ingredients, utensils, and any other restockable, purchasable item.

- Attributes:
  - **ID**: A unique integer representing this item. Used for quick reference by the database and other systems
  - **name**: The name of the item in question. Examples: “Short Grain White Rice”, “Napkin”, “Arugula”
  - **category**: What kind of classification the item falls under. Classifications are Food, Beverage, or Miscellaneous. Miscellaneous encompasses non-edible items like utensils and napkins.
  - **quantity**: How many items the restaurant has in storage.
- Operations:
  - **getQuantity**(): returns the quantity of the item
  - **setQuantity**(): sets the quantity of the item to an inputted number
  - **getName**(): returns the name of the item
  - **setName**(): changes the name of the item to an inputted name
  - **getID**(): returns the id of the item
  - **setID**(): sets the id of the item to an inputted integer
  - **remove**(): removes the item from database and storage

Inventory Processing:

The system that processes the quantity of inventory items available. It interacts with the database to receive information about availability of inventory items. It provides information about how many menu items can be created with the current stock.

- Operations:
    - **getItemByID**(int): Given an ID, fetches the corresponding inventory item from the database and returns it. Allows for faster querying compared to searching by name.
    - **getItemByName**(String): Searches for the inventory item with a given name in the database
    - **orderMenuItem**(MenuItem, int): Will change the count of items in the inventory database after a particular Menu Item has been created. For example, if "Sushi" is ordered, will recalculate the amount of rice, fish, seaweed, and other ingredients in storage.
    - **quantityCanMake**(MenuItem): returns the total number of Menu Items that can be made based on how many ingredients are in the inventory database. For example, if "Sushi" is inputted and there is only enough rice for 5 Sushi rolls, then the function returns 5.

Worker:

This class represents any worker in a particular restaurant.

- Attributes:
    - **username**: a unique string attribute used to represent the user's username when logging into the system through the login system
    - **name**: A string that represents the name of the user
    - **age**: An integer that represents the age of the user
    - **position**: A string that represents the position of the worker (Manager, server, hostess, chef)
    - **timeWorked**: A floating point value that represents how long the user has worked in a given time frame
    - **email:** The worker's email address. They provide this when they create their account.
- Operations:
    - Any worker in our system can use two operations
        - **sendOrder**(MenuItem[]): a function that sends an order to every user in the system so that everyone can view a customer's order.
            - The purpose of this function is mainly for seamless communication between chefs and servers. Chefs can view the orders of tables taken by the server. It also allows anyone to view a table's order so that delivering food to a table becomes easier.
        - **updateStock**(InventoryItem): a function that allows the user to update the quantity of certain items
            - The purpose of this function is to update the stock so that staff won't be misinformed about an item is in stock and make the correct adjustments when informing a customer
        - The worker class acts as a superclass to two subclasses:
            - Managers have certain operations that a regular worker does not have

        - **voidItem():** This function can void any item that has been already ordered
            - The purpose is if a user sends an order by mistake, the manager can void the item so the customer won't be charged.
        - **refundItem**(): This function can refund an item that has already been purchased
        - **shiftTimeWorked:** This function allows for the manager to change the time clock of a worker so that their hours worked can be correctly calculated when calculating hourly pay.

Menu Item:

This class represents the objects stored inside the Inventory Database.

- Attributes:
    - **id**: A unique integer value that is associated with a specific product. Users can access Inventory objects either by getting the id or name.
    - **name**: A String associated with a specific product used to access Inventory items from the Inventory database.

Transaction:

This class represents the object stored inside the Transaction Database.

- Attributes:
    - **Type**: A string that notates what payment type used to fulfill the payment requirement. This could be cash, debit/credit, or some 3rd party payment a**pp.**
    - **Amount**: A float that is the amount paid for the specific transaction.
    - **orderNumber**: An integer that associated with the specific transaction
    - **items**: This in an array of Strings in the with each String having the below format:
        - "ID: number purchased"
    - **date**: A Date object that reflects the time of purchase.
    -
- Operations:
    - **getInfo()**:This will return a string representation of the values held by the members of an inventory object. This method will be used by the user when trying to refund a customer.

Transaction System:

The system that interacts with the Transaction Database to process and provide relevant information about restaurant-customer and restaurant–supplier transactions. A transaction is defined as an exchange of money for an item.

- Operations:
    - **handlePayment**(Transaction): adds the provided Transaction to the Transaction Database. Attempts to process the payment between each party involved. More arguments may be added depending on how the payment is processed.
    - **updateRefund(**Transaction): Attempts to process a refund between each party involved in a given Transaction.

        - **getTransactionReceipt**(Transaction): returns a String that represents the text to be printed on a receipt of the transaction. This will largely be used for restaurant-customer transactions.

Login System:
The system that validates restaurant staff logins. Interacts with the Employee Database to receive and send information about workers' accounts, usernames, and passwords.

- Operations:
    - **validatePassword**(String user, String pass): The first string is the username of an employee's account. The second string is a password. This function returns whether or not the inputted password is the correct one for the user's account.
    - **forgotPassword**(String username): Sends a form to the inputted user's email to reset their password.
    - **getWorker**(String username): Returns the Worker object representing an employee's account. Used to access their account information.

Development Plan and Timeline

Start Date: November 1st 2024
End Date: December 1st 2024

The time frame being allocated for this project is approximately one month. The initial two weeks will consist of team members working on their individual projects or components. The following list shows the divvying up of tasks amongst team members:

Cody ⇒ Inventory: This encompasses creating the Inventory Objects and associated function to interact with the database.
Andrew ⇒ Login: Creation of the login authentication component that interacts with an employee database to ensure workers can access the overall system.
Husain ⇒ Payment: Working with external third party system to accept transactions and track info through Transaction objects stored in the Transaction DB.

After doing unit testing and such to ensure our respective components are working, the second two weeks are going to be integration and system testing. During this time, team members will be collaborating actively to ensure seamless integration of components for the final deliverable.