WolfWR Database Management System

CSC 540 Project Report 1

Matt Farver, Nick Garner, Jonathan Nguyen, Wyatt Plaga

Project Assumptions and Modifications

1. Products have a single, unique supplier - e.g. all bananas come from the same supplier
2. Products can only have one discount active at a time
3. Transaction, staff, and member IDs are unique across all stores
4. A single Warehouse processes all Supplier shipments for the entire chain, and is represented as a subclass of Store
5. rewardAmount and amountOwed are additional attributes attached to Member and Supplier entities, respectively, to keep track of the money to be paid to each as shipments and transactions occur.

Problem

We are designing and building a database system for WolfWR, a WolfCity wholesale-store chain. The chain will need a database system to maintain all the information of the warehouse. It will maintain information for all stores, staff, merchandise, club members, sign-ups, transactions, suppliers, and discounts for certain products. With the database being used by the staff of the store, each user will have different operations to maintain the database. There will be many different operations that the database will maintain such as process information, generate reports, maintain inventory, billing, and transaction records.

The purpose of using a database over other methods to maintain information is that a database is able to efficiently handle and reduce the redundancy in storing information. When the database relationships are properly established, it allows data to be properly organized in order for multiple users to query, sort, and manipulate the data in a timely manner.

Intended Classes of Users for System

1. Registration Operators
    a. Registration Operators will be able to sign up, renew, or cancel a customer's membership to the warehouse store. They will have access to the staff information, member information, and store information.
2. Warehouse Operators
    a. Warehouse Operators will be able to work directly with the supplier and maintain the products of each store. This means that they are able to order more products from the supplier, transfer products from store to store, and handle returns. These operators will have access to the store information, supplier information, and merchandise information.
3. Billing Operators
    a. Billing Operators will be handling the reports and billing that is required between the supplier and the member. They are responsible to send payments for what the supplier is owed and to send yearly rewards that members get from buying certain products. The billing operators will have access to member information, and supplier information.

Five Things To Keep Track Of

The database needs to enable the store to keep track of:

1. **What** is being sold. This means keeping information regarding the merchandise, such as the product ID, name, quantity in stock, buy price, market price, production data, and expiration date. In addition, information regarding discounts will need to be kept.
2. **Where** it's being sold. This means keeping track of information regarding the stores, such as the store ID, manager ID, store address, and phone number.

3. **Who** is buying the product. This means keeping track of member information such as a customer's id, their first and last name, membership level, email address, phone, home address, and activity status. This also means keeping track of sign-up information, such as the store id, sign-up date, staff id, and customer id.
4. **How** their product is being bought. This means keeping track of information about their suppliers, such as supplier id, supplier name, phone, email address, and location.
5. **How** their product is being sold. This means keeping track of transaction information, such as transaction ID, store id, customer id, cashier id, purchase date, product list, and total price.

Operation Examples

Example #1 - One situation where an operation might need to occur would be if a store runs out of a product and requests a transfer from another store. When the warehouse operator initiates a transfer between two stores, this would require *maintaining inventory records.*

Example #2 - Another situation where an operation would need to occur when a registration operator signs up a member. *Information processing* would need to occur as the enter a new sign-up into the database.

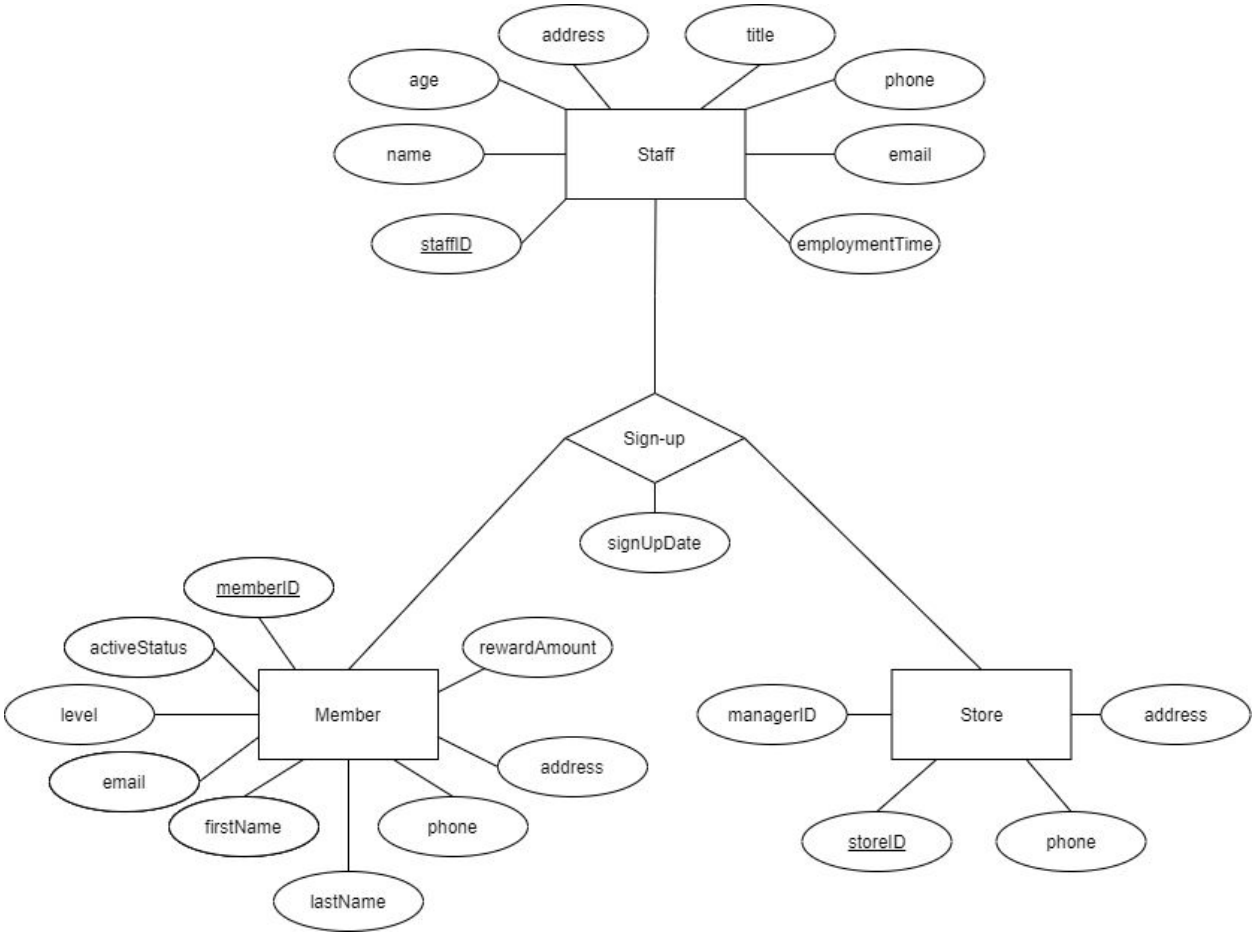
Operation APIs - Included at the end due to length

Views of the Data

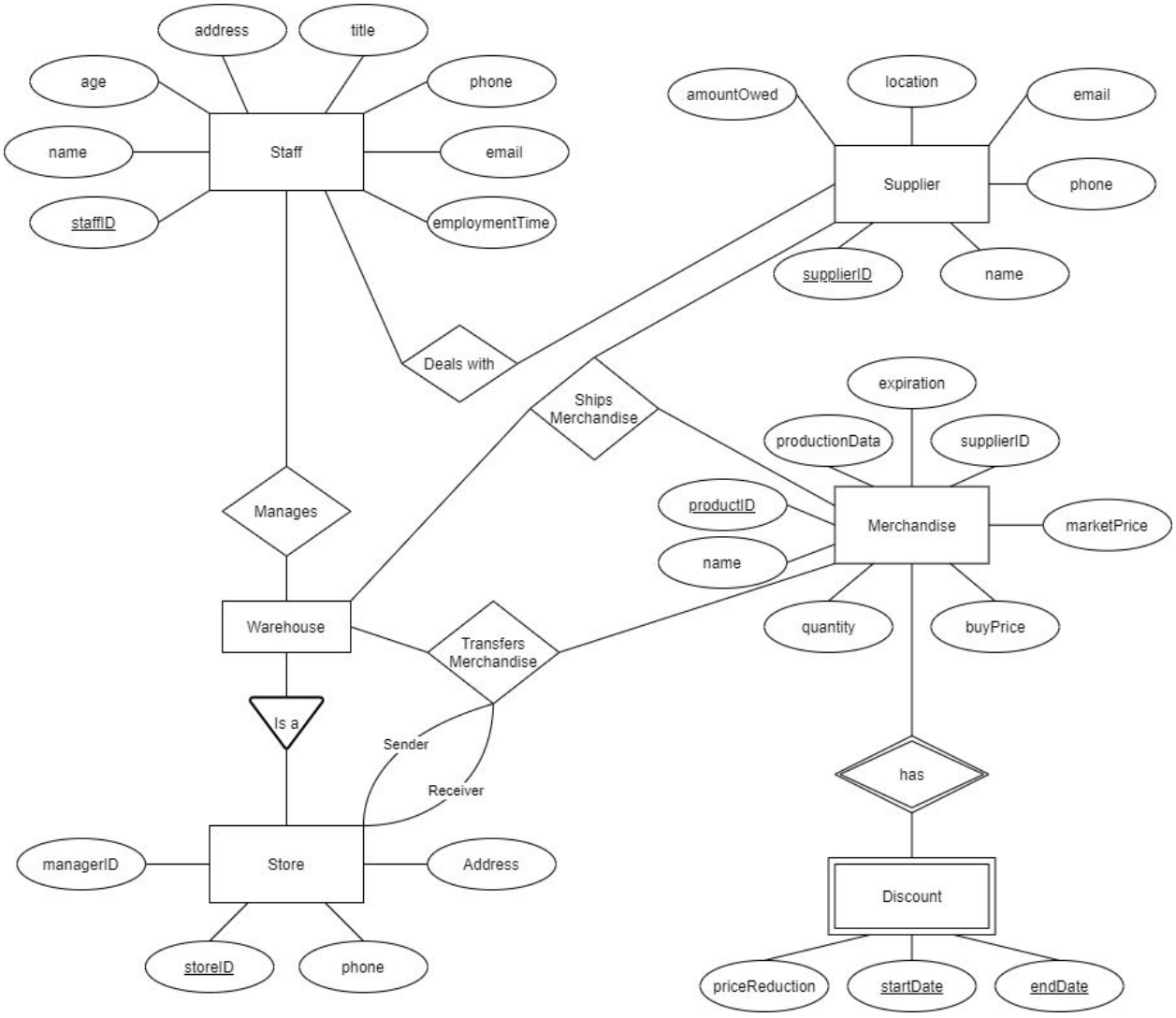
1. Registration Operator
    a. To manage the lifecycle of a member, registration operators will have views to all the data for store staff, members, and store information. This includes member sign-up data such as the date, and store where the member signed up.
2. Warehouse Operator
    a. Warehouse operators will need views into all the data for suppliers, stores, and merchandise including the merchandise discount information to perform their responsibilities. Key data attributes for warehouse operators include, Merchandise quantity, merchandise discount, and store IDs and store addresses.
3. Billing Operator
    a. Billing operators will require views into all membership data, transaction data, and supplier data to perform their required operations. Key data attributes for billing operators include, membership reward amounts, members active status, supplier amount owed, and all transactional data.

Local Entity-Relationship Diagrams

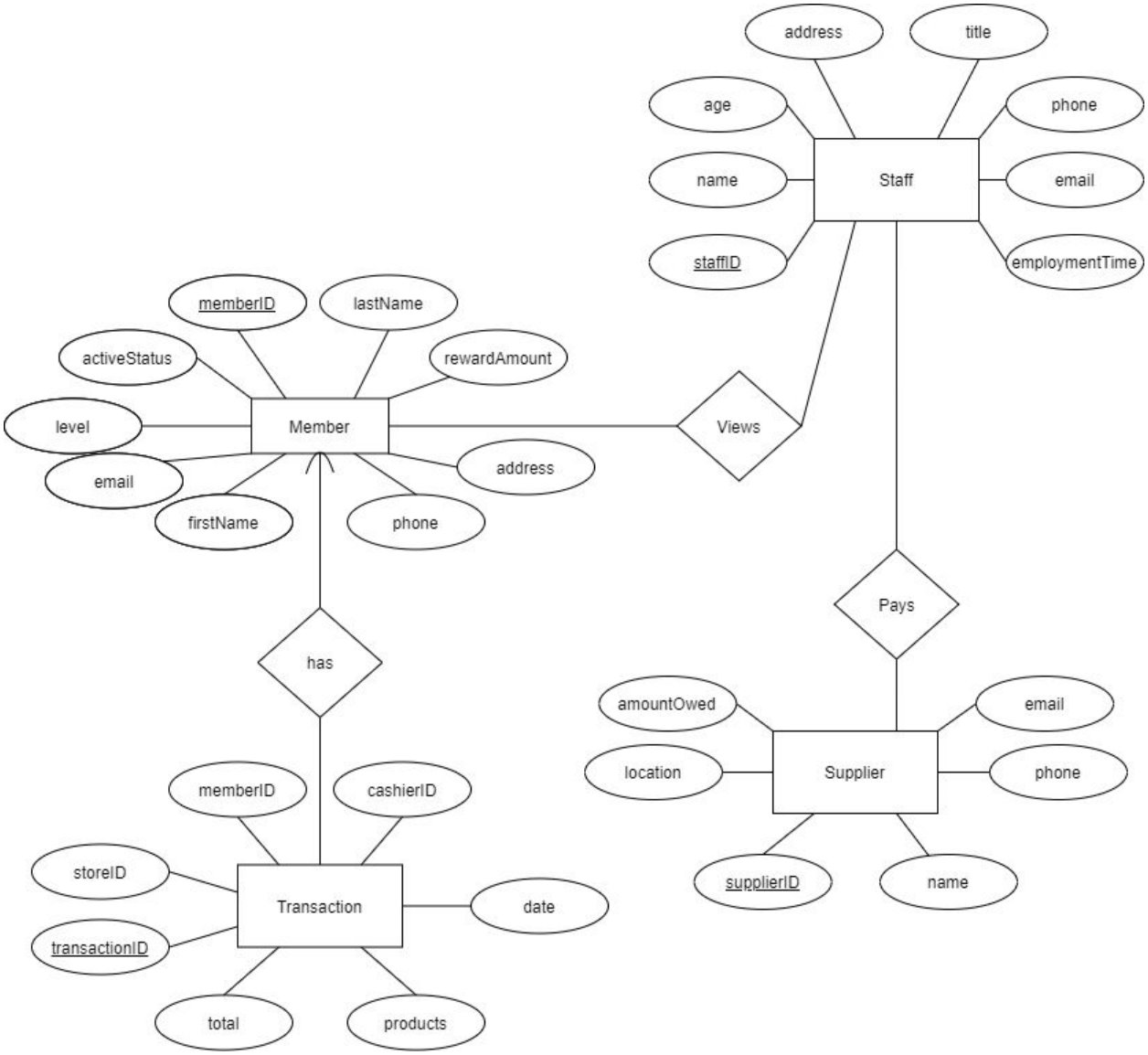
Registration Operator



Warehouse Operator



Billing Operator



Local E/R Diagram Documentation

1. User Classes
    a. The user classes that will be using the database system will be the Registration Operators, Warehouse Operators, and the Billing Operators. Each user has their own ER diagram in the section above.

2. Entity Sets
    a. Staff
        i. The staff entity set can be a manager, cashier, registration operator, warehouse operator, billing operators, and others. This entity set was created in order to reduce redundancy and to be able to contain the common attributes of all staff. Such as staffID, name, age, address, title, phone number, email, and employment time.

    b. Member
        i. The member entity set created to contain the common attributes of all members. Such as memberID, active status, level, email, first name, last name, phone, address, and reward amount.

    c. Store
        i. The store entity set was created since this database is for a warehouse chain which means that there will be many stores in different locations. The attributes that will be stored in this entity set are the mangerID, storeID, phone, and address.

    d. Supplier
        i. The supplier entity set was created since there will be many different suppliers that the stores will be buying products from. The attributes that will be stored in this entity set are the supplierID, name, phone number, email, location, and amount owed.

    e. Merchandise
        i. The merchandise entity set was created since there will be many products in the store each with different information to maintain. The attributes of the products that will be maintained are the productID, name, production data, expiration date, supplierID, market price, buy price, and quantity.

    f. Transaction
        i. The transaction entity set was created as there will be many transactions and rewards will be paid to a member depending on each transaction. This entity will have attributes such as the transactionID, memberID,

cashierID, storeID, products, total amount, and date.

g. Discount
    i. The discount entity set is a weak entity set. This entity set will depend on the existence of a merchandise. The entity will hold attributes such as the price that the product is reduced, the day that the product goes on sale, and the day that the product's sale ends.

3. Relationships
    a. Is a
        i. The chain's single central Warehouse is represented in the database as a subclass of Store, with no additional attributes.
    b. Sign up
        i. A three-way relationship between connecting a newly created Member with the Store in which they joined the chain and the Registration Operator that processed their membership.
    c. Deals with
        i. A Warehouse Operator (Staff) coordinates Merchandise orders with the relevant Supplier.
    d. Ships Merchandise
        i. Suppliers ship Merchandise to the central Warehouse for processing and distribution.
    e. Transfers Merchandise
        i. Warehouse sends Merchandise to Stores, and Stores can transfer Merchandise between themselves once received.
    f. Has
        i. (Warehouse Operator) Merchandise may have multiple price Discounts but only one can be active at a time - i.e. no overlapping dates.
        ii. (Billing Operator) Transactions belong to exactly one Member
    g. Views
        i. Billing Operators can view Member information to calculate rewards for platinum members.
    h. Pays
        i. Billing Operators pay Suppliers based on their amountOwed as calculated by the Warehouse Operator when receiving a shipment. This value is then reset after payment is sent.

4. Primary Keys
    a. staffID
        i. The staffID will be used to associate a transaction and which staff member signs up a customer.

    b. memberID

i. The memberID will be used to pull up information about how many rewards they can receive at the end of the year and if their membership is renewed or expired.

c. storeID
    i. The storeID will be used to sign up a customer to a store, and sending products to a store.

d. productID
    i. The productID will be used by the Warehouse Operator to update the quantity of products or add new products to the store.

e. supplierID
    i. The supplierID will be used by the Billing Operator and the Warehouse Operator when payments are being made or products are being handled.

f. transactionID
    i. The transactionID will be associated with a member and how many rewards they are accruing throughout the year.

Local Relation Schemas and Documentation

Registration Operator

Member(memberID, firstName, lastName, address, phone, email, level, activeStatus, rewardAmount)
Store(storeID, managerID, address, phone)
Staff(staffID, name, age, address, title, phone, email, employmentTime)
SignUp(memberID, storeID, staffID, signUpDate)

The E/R diagram for the registration operator view contained four entity sets and one three-way relationship. The relation for each entity set was created in a straightforward manner, in which the key(s) was listed first and underlined, followed by all of the attributes associated with that entity group. “SignUp” is a three-way relationship, and so the three keys from the participating entity groups were listed, followed by the single attribute, “signUpData,” associated with the relationship itself. The Registration view is fairly simple and really only involves signing up new members, thus only the Staff, Store, and Member entities were included.

Warehouse Operator

Store(storeID, managerID, address, phone)
Staff(staffID, name, age, address, title, phone, email, employmentTime)
Supplier(supplierID, name, phone, email, location, amountOwed)

Merchandise(productID, supplierID, name, quantity, buyPrice, marketPrice, expiration, productionData)
Discount(productID, startDate, endDate, priceReduction)
Deals with(supplierID, staffID)
Ships Merchandise(supplierID, productID, storeID)
Transfers Merchandise(sendingID, receivingID, warehouseID, productID)

The E/R diagram for the warehouse operator view contained six entity sets and four relationships. The relationship schema for each of the five strong entity sets was created in a straightforward manner, in which the key was listed first and underlined, followed by all of the attributes associated with that entity group. The “discount” weak entity set was converted to a relationship schema by first listing the key of its supporting entity set, “merchandise,” followed by the attributes of the “discount” entity set itself. Being a supporting relationship, the “has” relation need not be converted at all. All of the non-is-a relationships, not having any attributes associated with the relationship themselves, were converted simply by including all of the keys of the entity sets which they connected. For instance, the “deals with” relationship connected the “supplier” entity group with the “warehouse operator” entity group, and so the key of each entity set was listed. Having no attribute associated with that entity set, no further attributes were listed. “Transfers Merchandise,” being a self-referential relationship, needed to include the key of the “Store” entity set twice, and so the key was given different names “sendingID” and “receivingID.” The “null-value” approach was used for treatment of the subclasses. Since the “Warehouse Operator” has no attributes not shared by the “Staff” entity set, the two entity sets were converted to a single “Staff” relationship schema.

Billing Operator

Member(memberID, firstName, lastName, address, phone, email, level, activeStatus, rewardAmount)
Supplier(supplierID, name, phone, email, location, amountOwed)
Transaction(transactionID, storeID, cashierID, date, total, products)
Views(memberID, staffID)
Pays(supplierID, staffID)
has(transactionID, memberID) [able to be combined with the relation for the transaction entity set as it is many-to-one]

The E/R diagram for the warehouse operator view contained five entity sets and four relationships. The relationship schema for each of the entity sets was created in a straightforward manner, in which the key was listed first and underlined, followed by all of the attributes associated with that entity group. The “has” relation, being many-to-one, need not be converted into a relationship. If this were done, the memberID key would be included in the “transaction” relation. The other three relations, none having any attributes associated with the relation themselves, were converted simply by including all of the keys of the entity sets which they connected. The “null-value” approach was used for treatment of the subclasses. Since the

“Billing Operator” has no attributes not shared by the “Staff” entity set, the two entity sets were converted to a single “Staff” relationship schema.

Information Processing - APIs

Store Information APIs

Get store information:

Input:

- Store ID - Optional - (Will return all stores if null)

Response:

- Null if data is not found
- Store ID, Manager ID, Address, Phone

Create a new store:

Input:

- Manager ID - Optional (Assuming a manager ID is not required to create a new store)
- Address - Required
- Phone -Required

Response:

- Returns confirmation
- Store ID Internally Created

Update store information:

Input:

- Store ID - Required
- Manager ID - Optional
- Address - Optional
- Phone - Optional

Response:

- Returns confirmation

Delete a store

Input:

- Store ID - Required

Response:

- Returns confirmation

Staff Information APIs

Get staff members

Input: Conditional

- Store ID - Optional (will return all staff if null)
- Staff ID - Optional (will return single staff member if present)

Response:

- Null if data is not found
- Staff ID, Store ID, Name, Age, Address, Phone, Email, Employment Time, Job Title

Create a staff member:

Input:

- Store ID - Required
- Name - Required
- Age - Required
- Address - Required

- Title - Required
- Phone - Required
- Email - Required
- Employment Time - Required

Response:

- Returns confirmation
- Staff ID Internally Created

Update a staff member:

Input:

- Staff ID - Required
- Store ID - Optional
- Name - Optional
- Age - Optional
- Address - Optional
- Phone - Optional
- Email - Optional
- Employment Time - Optional

Response:

- Returns confirmation

Delete a staff member:

Input:

- Staff ID

Response:

- Returns confirmation

Member Information APIs

Get club member information

Input: Conditional

- Store ID - Optional (Will return all store customers if Null)
- Member ID - Optional (Will return all customers if Null)
- Level - Optional (“Gold” or “Platinum”)

Response:

- Null if data is not present
- First Name, Last Name, Level, Email, Phone, Address, Active Status, Sign-up Date, Reward Amount

Create a member (Sign-up a Member)

Input:

- Store ID - Required
- Staff ID - Required
- First Name - Required
- Last Name - Required
- Level - Required
- Email - Required
- Phone - Required
- Home - Required

Response:

- Returns confirmation
- Member ID internally created, Status == “Active”, Sign-up Date == “current date”, Reward Amount == “0”

Update a member:

Input:

- Member ID - Required
- First Name - Optional
- Last Name - Optional
- Level - Optional
- Email - Optional
- Phone - Optional
- Address - Optional
- Sign up Date - Optional
- Reward Amount - Optional
- Active Status - Optional

Response:

- Returns confirmation

Delete a customer:

Input:

- Member ID - Required

Response:

- Returns confirmation

CANCEL customer membership API:

Input:

- Member ID - Required

Response:

- Returns confirmation
- Active Status == “Inactive”

Supplier Information APIs

Get a supplier:

Input:

- Supplier ID - Optional

Response:

- Null if data is not found
- Name, Phone, Email, Location, Amount Owed

Create a supplier:

Input:

- Supplier Name - Required
- Supplier Phone - Required
- Supplier Email - Required
- Supplier Location - Required

Response:

- Returns confirmation
- Supplier ID internally created, Amount Owed == “0”

Update a supplier:

Input:

- Supplier ID - Required
- Supplier Phone - Optional
- Supplier Email - Optional
- Supplier Location - Optional
- Amount Owed - Optional

Response:

- Returns confirmation

Delete a supplier:

Input:

- Supplier ID

Response:

- Returns confirmation

Maintaining Inventory Records - APIs

Merchandise Information APIs - Maintaining Inventory

Get merchandise information

Input: Conditional

- Product ID - Optional (will return all products if Null)
- Store ID - Optional (Will return product info from warehouse if Null, otherwise it will return store product information

Response:

- Returns all product information from the warehouse unless store id is provided
- Product ID, Name, Quantity, Buy Price, Market Price, Production Data, Expiration Date, Supplier ID, Price Reduction, Start Date, End Date

Create merchandise

Input: Assumption - All new products created by the warehouse before going to a specific store

- Name - Required
- Quantity - Required
- Buy Price - Required
- Market Price - Required
- Production Data - Required
- Expiration Date - Required
- Supplier ID - Required
- Price Reduction - Optional
- Start Date - Optional
- End Date - Optional

Response:

- Returns confirmation
- Product ID internally created, Price Reduction == “0”, Start date and End date == “null”

Update merchandise for including sales information

Input:

- Product ID - Required
- Store ID - Optional (If not provided, it will update warehouse inventory)
- Quantity - Optional
- Buy Price - Optional
- Market Price - Optional
- Production Data - Optional
- Expiration Date - Optional
- Supplier ID - Optional
- Price Reduction - Optional
- Start Date - Optional
- End Date - Optional

Response:
- Returns confirmation

Delete a product from a store

Input:
- Product ID - Required
- Store ID - Optional

Response:
- Returns confirmation

TRANSFER merchandise API: Transfer product from one store to another

Input:
- Product ID - Required
- Sending Store ID - Required (can be warehouse or another store)
- Receiving Store ID - Required
- Quantity - Required
- Buy Price - Required
- Market Price - Required
- Production Data - Required
- Expiration Date - Required
- Supplier ID - Required
- Price Reduction - Optional
- Start Date - Optional
- End Date - Optional

Response:
- Returns confirmation
- If merchandise does not exist in receiving store, it will be created
- Inventories are updated in sending and receiving stores

RETURN merchandise API: Product returns that update inventory and transaction

Input:
- Transaction ID - Required
- Returned Product list IDs - Required
- Store ID - Required

Response:
- Returns confirmation
- Transaction is updated and product inventory is updated

Maintaining Billing & Transaction Records - APIs

Transaction Information APIs

Get a transaction:

Input: Conditional given the inputs provided
- Transaction ID - Optional
- Store ID - Optional
- Member ID - Optional

Response:
- Store ID, Member ID, Cashier ID, Date, Products, Total

- May return all transaction given the provided input

Create a transaction:

Input:
- Store ID - Required
- Member ID - Required
- Cashier ID (staff id for cashier) - Required
- Date - Required
- Products - Required
- Price - Required

Response:
- Returns confirmation
- Transaction ID - internally created

Update a transaction:

Input:
- Transaction ID - Required
- Member ID - Optional
- Cashier ID (staff id for cashier) - Optional
- Date - Optional
- Products - Optional
- Price - Optional

Response:
- Returns confirmation

Delete a transaction:

Input:
- Transaction ID

Response:
- Returns confirmation

CREATE a bill to pay suppliers:

Input:
- Supplier ID

Response:
- API will reset the amount owed
- Supplier Name, Supplier Phone, Supplier Email, Supplier Location, Amount Owed

CREATE reward checks for Platinum members:

Input:
- Member ID - Optional (will create reward checks for all platinum members if null)

Response:
- Returns reward checks for Platinum members and resets the reward amount to zero
- First Name, Last Name, Level, Email, Phone, Address, Reward Amount

---

Generate Reports APIs

Sales Reports

GENERATE total sales reports for stores:

Input:
- Start Date - Required
- End Date - Required
- Timing Breakout - Required (Day, Month, Year)

- Store ID - Optional - will return all stores if null

Response:

- Generates a report displaying total sales for all stores for a given period of time or for a single store for a given period of time.

GENERATE sales growth report for a store during a period of time (Month over Month sales growth):

Input:

- Store ID - Required -
- Start Month/Year - Required
- End Month/Year - Required

Response:

- Generates a file displaying the MoM sales growth by store or for the entire chain

GENERATE product inventor report per store:

Input: Conditional

- Store ID - Optional (will return all stores if Null)
- Product ID - Optional (will return all products if Null)

Response:

- Will generate a report with all product information for a given store, or for a specific product.

GENERATE member growth by month:

Input:

- Start Month/Year - Optional (if Null will start with first month data is available)
- End Month/Year - Optional (if Null will end with the last completed month)
- Store ID - Optional (will return customer growth for specific store if provided)

Response:

- Will generate a report with MoM member growth for the chain or a for a specific store

Member ACTIVITY report:

Input:

- Customer ID - Required
- Start Date - Optional (if Null, will start with customers first transaction)
- End Date - Optional (if Null, will end at the current date)

Response:

- Will generate a report with member activity during a specified period of time