CS348 Project

Milestone 2

Vincent Chen

Michael Huang

Chong Luo

Muhammad Salman

# Task 1

The application we are building manages the inventory of a convenience store. The idea of the app is that the owner of the shop can add or delete items to the database with price, quantity available and it tells you what items are on sale by what amount. The customers can use the app to find the items, prices and if the quantity is available. A brief description of the application features is provided below. More details are provided in the files available in the code folder.

Dataset that we are going to be using will directly be obtained from the owner of the convenience store that we are building the app for. This dataset will include all the items that are sold in the store. The dataset includes the the following attributes of the items:

* Unique Id
* Name
* Price
* Stock (Quantity of the item available)
* Sale Tag (Identifies if the item is on sale, the percent off, and the expiration date)

Furthermore, items are divided into subcategories, each of which has a unique id and name attribute. The application has to serve two target audiences. The first is the owner of the store, and the interface is as follows:

1. Add new items to the Database
2. Update item (price, stock, subcategory)
3. Delete item from Database
4. Add/delete new categories/subcategories
5. Put a sale tag on an item
6. Remove expired sales

The owner’s use case is to modify the database from a high level. To accommodate this, the owner will be distinguished from regular customers during login (this is omitted as a feature because it’s lumped in with the “shop as a registered user” feature for customers).

The second target audience is the customers, and the application has to be capable of the following:

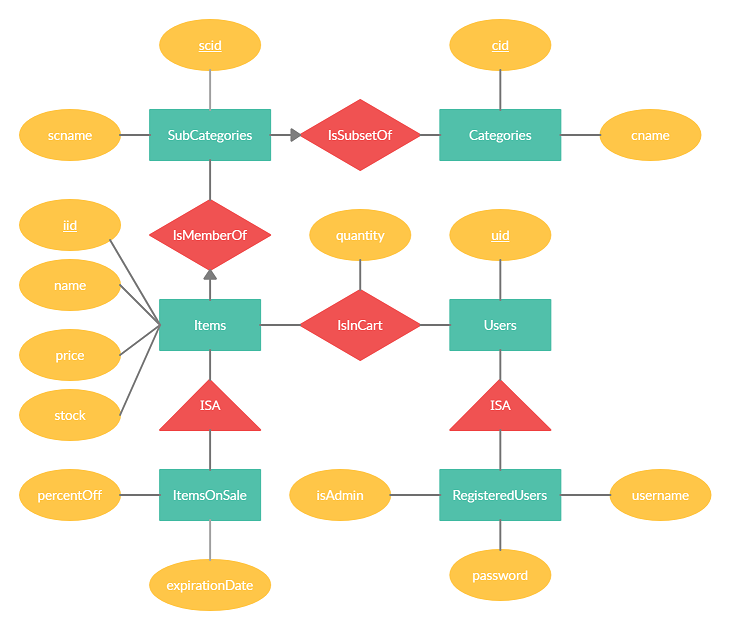
1. Search item by name
2. Get the items on sale
3. View items by categories
4. Shop as a registered user (with username/password)
5. Shop as a temporary user (without username/password)
6. Add item to cart
7. Clear customer cart (upon checkout)

# Task 2

**Assumptions about data:**

* Item IDs, Category IDs and SubCategory IDs are unique
* User IDs and usernames are unique
* Price and percentOff are nonnegative floats, stock is a nonnegative integer
* expirationDate is sometime in the future

**E/R Diagram**



* We have three main entity sets: Items, Categories, SubCategories and Users.
  + Items are the items for sale.
  + Categories are the various types of products that can be sold (e.g. Food/Drink, Accessories, etc.)
  + SubCategories are more specific categories (e.g. Cheese and Milk are subcategories for Dairy)
  + Users are the customers, who are capable of adding items to cart and checking out. Each user has a unique uid
* The IsMemberOf relationship signifies that an item belongs to a subcategory. This relationship is one-to-many, since each subcategory can store an arbitrary number of items and each item can belong to at most one suitable subcategory.
* The IsSubsetOf relationship signifies that a subcategory is a subset of (or more specifically belongs to) a category. This relationship is one-to-many, since each subcategory belongs to at most one category, but each category can accommodate multiple subcategories.
* The IsInCart relationship signifies that an item is in a cart. This relationship is also many-to-many, since each item can be in multiple users’ carts, and every user can have multiple items in their cart. It includes a “quantity” attribute, so a customer can purchase several copies of the same item.
* We use a subclass/superclass relation to store items on sale, with an entity-in-all-superclasses approach.
  + This is so we don’t have to waste space storing for each item whether or not it is on sale (for instance, with some onSale boolean attribute).
  + This may change later if we expect to see a large portion of items on sale.
* We use a subclass/superclass relation to store registered users, with an entity-in-all-superclasses approach
  + Registered users differ from “guest” (temporary) users in that they have a username and password and can login, as well as have an isAdmin field to allow the owner (or other authority figures) to manage the database.
  + Guest users are assigned a temporary unique uid so they can add items to their cart, which is removed after they finish shopping.

**Database tables**

* Items(iid, name, price, stock)
* Categories(cid, cname)
* SubCategories(scid, scname)
* Users(uid)
* IsMemberOf(iid, cid)
* IsSubsetOf(cid, scid)
* IsInCart(iid, uid, quantity)
* ItemsOnSale(iid, percentOff, expirationDate)
* RegisteredUsers(uid, username, password, isAdmin)

# Task 3

Items

|  |  |  |  |
| --- | --- | --- | --- |
| **iid** | **name** | **price** | **stock** |
| 0 | Chips | 1.00 | 100 |
| 1 | Soda | 1.00 | 100 |
| 2 | Dental floss | 2.00 | 20 |
| 3 | Lottery ticket | 1.00 | 999 |
| 4 | Tide Pods | 5.00 | 0 |

Categories

|  |  |
| --- | --- |
| **cid** | **cname** |
| 0 | Food/Drink |
| 1 | Household items |
| 2 | Miscellaneous |

SubCategories

|  |  |
| --- | --- |
| **scid** | **scname** |
| 0 | Food/Drink |
| 1 | Household items |
| 2 | Miscellaneous |

Users

|  |
| --- |
| **uid** |
| 0 |
| 1 |
| 99 999 999 |

IsMemberOf

|  |  |
| --- | --- |
| **iid** | **cid** |
| 0 | 0 |
| 1 | 0 |
| 2 | 1 |
| 3 | 2 |
| 4 | 0 |
| 4 | 1 |

IsInCart

|  |  |  |
| --- | --- | --- |
| **iid** | **uid** | **quantity** |
| 0 | 0 | 2 |
| 1 | 0 | 1 |
| 3 | 0 | 100 |
| 2 | 99 999 999 | 1 |
| 3 | 99 999 999 | 2 |

ItemsOnSale

|  |  |  |
| --- | --- | --- |
| **iid** | **percentOff** | **expirationDate** |
| 0 | 20.00 | 2021-07-01 |
| 1 | 10.00 | 2021-08-24 |

RegisteredUsers

|  |  |  |
| --- | --- | --- |
| **uid** | **username** | **password** |
| 0 | John Deer | password123 |
| 1 | Dear John | 123password |

# Task 4

The UI we choose is a command line application developed through visual studio c++ project. See the solution folder in code.zip. A readme.md file is also provided.

# Task 5

SQL queries in ProjectMilestone1SampleSQL.sql.

# Task 6

We use MySQL as backend and c++ as front end. Please refer to readme.md in the code.zip for detail information.

# Task 7

The intent of this project is to build a DBMS and a front-end UI for a convenience store and its users to navigate it. Our original plan was to obtain a real dataset from a store Chong’s uncle owns in Shanghai. However, due to logistical challenges, his uncle was unable to provide the dataset in the timeframe required for Milestone 2.

Hence, we chose to use a public grocery store dataset from online.

It can be obtained in the following link: <https://www.kaggle.com/jeffryjames/groceries-data>

This dataset is scrapped from Jiomart, a real grocery store from India, and has over 2200 items entries. This should more than suffice for general and performance testing for Milestone 2.

We would need to parse the data from the excel sheet and add unique item IDs to each one of these items. Though this is easy, as each row entry is one unique item, we can assign the row number to each item as the item id.

# Task 8

Looking over the dataset, we immediately recognized that having just one layer of categories was not going to be enough for over 2200 items. A single query of a category could return hundreds of items. This would be a performance issue. So we added sub categories to add another layer of organization to items. This means that when a user queries by category, they first have to select a general category and a sub category before the query returns a list of items. This reduces the return size of the list of items by a large amount.

Another problem we ran into was how to implement and distinctly identify the owner or admins of the store versus regular users. Our schema from Milestone 1 could not accommodate separation of privileged operations such as modifying the database entries from unprivileged operations such as an item look up. So we modified the schema to allow for guest users, registered users, and admin users.

# Task 9

Please refer to code.zip for the application and sample outputs.

# Members

|  |  |
| --- | --- |
| **Member** | **Effort/Progress** |
| Vincent Chen | Task 6, Task 4 |
| Michael Huang | Updates to Task 1, Task 2 |
| Chong Luo | Task 5 |
| Muhammad Salman | Task 1 |