CS166 Project Report

Project Summary

CS166 Project by Patrick Liu and Claudia Pascal

The Retail System is a system designed for a retail chain of shops to manage inventory and invoices across locations.

This report is an outline of our work for the project. To clarify any ambiguity within this document, please refer to the source code for what is actually implemented.

Create Tables, Bulkload Data Scripts

- `start_db.sh` and `stop_db.sh` are the same as the scripts provided for labs.
- `create_db.sh` was modified to include `triggers.sql` .
  - provided that it runs from the `script` directory, it also changes directories and loads test data into the database

Source Code

- no particular requirements are needed to compile and run this code.
  - the GUI, however has some requirements (outlined in `README.md`)

Other scripts like triggers, stored procedures, and indexes

- indexes are included in the `create_indexes.sql` file under `project/sql/src`.
  - In the same directory, `triggers.sql` contains some triggers used. Details are outlined in the section below.

Extra Credit

Good User Interface

- All user input is sanitized
  - Users cannot order from a store they're not in range of
  - Managers cannot perform manager actions on stores they don't manage
  - All product names are checked to confirm the product exists
- Orders are shown with most recent at bottom

Graphical User Interface (Web Browser)

- a simple GUI was made using Django. It allows for viewing a list of all stores and each store's products. It also displays a list of all users in the system. No other functionality is provided through this GUI.
    - details for running the GUI are outlined in `README.md`.

Triggers and Stored Procedures

- three triggers were incorporated into this project. More details are outlined in the `Details` section below.

Performance Tuning

- four indexes were incorporated into this project. More details are outlined in the `Details` section below.

Details

Indexes

`store_product_list`

- index for each store's product list
- `Product` table is large as products are unique to each store

`user_orders`

- index for each customer's orders
- while a customer may not have many orders, we could have a large table of `Orders`

`store_orders`

- index for each store's orders
- more efficient when a Manager wants to view a store's orders
    - this is a procedure that would most likely be needed in real world contexts. For example, a store that needs to create a quarterly financial report would find it handy to be able to quickly pull a store's orders.

Triggers

`ordernumber_trigger`

- keeps track of last used `OrderNumber` so we can quickly insert a new order

`supplyordernumber_trigger`

- keeps track of last used `RequestNumber` so that we can quickly insert a new supply request

`productUpdatenumber_trigger`

- keeps track of last used `updateNumber` so that we can quickly insert a new product update record

`updateaftersupplyorder_trigger`

- runs after a supply order is placed by a manager
- updates a store's inventory to include the supply order

JDBC Usage

Assumptions made about the project are outlined below.

General Usage

Anybody is allowed to create a user using the `CreateUser()` function to access the Retail System. A user consists of a username, password and their location (latitude, longitude).

A user must be logged in to view stores and their product offerings. This can be done by using the `LogIn()` function.

Users are allowed to browse stores and their product offerings through the use of `viewStores()` and `viewProducts()`.

In addition, a user can place an order by using `placeOrder()`. An order will only be placed if the store is within 30 miles of the user's location and the requested quantity is in stock at that store.

Users are allowed to see their five most recent orders by using `viewRecentOrders()`.

Manager Usage

Managers are a type of user. Managers are able to update a product's price and quantity in stock using `updateProduct()` only if they are the manager for the store.

Managers are allowed to view the five most recent updates to a store's product using `viewRecentUpdates()` only if they are the manager of that store.

Managers can view all the orders for a store they manage.

Managers can place product supply requests using `placeProductSupplyRequests()` only if they are a manager for that store.

Admin Usage

Admins are allowed to view and update all product information. However, no add or delete functionality is present in the system.

Admins are also allowed to update all user information. However, no add or delete functionality is present in the system.

Code Documentation

This section provides some detail about the functions we wrote. The descriptions are short to give an idea of how they are being used overall. Ultimately, the code should serve as its own documentation. Please refer to `Retail.java` for any ambiguity.

Helper Functions

`validProductSelection(Retail, String)`

- checks to see if the string is a product that exists in the `Product` table

`managerSelectStore(Retail, List<List<String>>)`

- given a list of stores, return the user's selection of a particular store
- empty string is a valid input

`userSelectStore(Retail, List<List<String>>)`

- similar to `managerSelectStore()` but meant for prompting non-admin/manager users
  - ensures a user selects a store from a given list
- does not accept empty string as valid input

`pickWarehouse(Retail)`

- select a warehouse from all warehouses available

`getNonEmptyResponse`

- ensures that a user's input is non-empty

`getResponse`

- returns a user's response (including empty string)

`isAuthorized`

- checks user's authorization level (customer, manager, admin)

Core Functions

```
MAIN MENU
---------
1. View Stores within 30 miles
2. View Product List
3. Place a Order
4. View 5 recent orders
Manager Functions:
---------
5. Update Product
6. View 5 recent Product Updates Info
7. View 5 Popular Items
8. View 5 Popular Customers
9. Place Product Supply Request to Warehouse
10. View All Orders Managers Manages
18. Admin View Users
19. Admin Update Users
.......................
```

The following functions correspond to the menu entries above. Each functions accepts a `Retail` object as a parameter, which is omitted from the headers below.

1) `viewStores()`

- displays stores within 30 miles of a user
- one mile is defined as 1 unit in the 100×100 grid system. This can be changed by adjusting the `ONEMILE` variable.

2) `viewProducts()`

- displays a store's products to the user
- no restrictions on viewing store's inventory to any user

3) `placeOrder()`

- place an order for a user
- allows orders only to stores within 30 miles of user
  - see `viewStores()` for more details
- ensures a user can only buy what a store has in stock

4) `viewRecentOrders()`

- display recent orders of a user

5) `updateProdut()`

- manager/admin only function
  - managers only allowed to update products of stores they manage
- fields that can be updated are `numberOfUnits` and `pricePerUnit`

6) `viewRecentUpdates()`

- manager/admin only function
  - managers can only view updates of stores they manage
- user can choose to display updates for all stores they have permission over

7) `viewPopularProducts()`

- manager/admin only function
  - managers can only see popular products for stores they manage
- user can choose to display the most popular products for all stores they have permission over

8) `viewPopularCustomers()`

- manager/admin only function
  - managers can only see the most popular customers for stores they manage
- user can choose to display the most popular customers at all stores they have permission over

9) `placeProductSupplyRequests()`

- manager/admin only function
  - managers can only place supply requests for stores they manage
- user must place supply request for item one at a time

10) `viewOrders()`

- manager/admin only function
  - managers can only view orders at stores they manage
- user can choose to display orders for all stores they have permission over

11) `viewUsers()`

- admin only function

- display all users

12) `updateUsers()`

- admin only functin
- edit user fields
  - allows for editing of all user fields
  - editing of `userID` cascades changes to respective user's orders as well