CG3002 Group 13 Software Report

Design

**Feature list**

HQ Admin:

* Login/logout
* Products
  + Create new product
  + View, search & sort list of products
  + Add product to outlet inventory
* Stocks
  + View, search & sort list of restock requests from all outlets
  + View detailed information of each request batch
  + Approve & forward request to suppliers
  + Accept goods delivered to online outlet
* Transactions
  + View, search & sort list of transactions from outlet selected
  + View detailed information of each transaction – barcode, quantity, price
* Inventory list of all outlets
  + View, search & sort inventory of all outlets
  + Edit inventory information
  + Discontinue a product from an outlet’s inventory
* Statistics
  + View pie chart of outlets’ revenue
  + View bar chart of daily outlet revenue over 7 day period
* Outlet
  + View, search & sort all outlets
  + Edit outlet information
  + Add new outlet – outlet name, location
* Online shop management
  + View, search & sort online transactions
  + View details of each online transaction
  + Dispatch an online transaction – sends email to customer

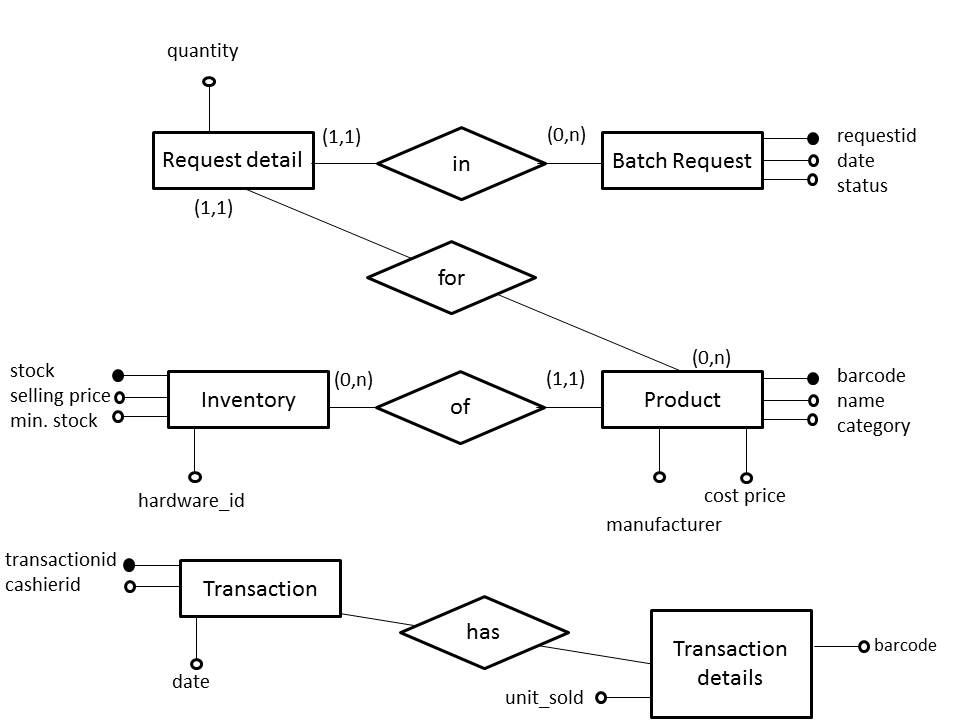
Shop PC:

* Inventory list of products in the outlet
  + View, search & sort inventory
* Price display
  + Add new price display and attach to a product
  + View list of price displays belonging to the outlet
* Stocks
  + View, search & sort list of restock requests from the outlets
  + View detailed information of each request batch
  + Accept goods delivered to the outlet
* Transactions
  + Add a new transaction
  + View, search & sort list of transactions from the outlet
  + View detailed information of each transaction – barcode, quantity, price
* Sync
  + Automatic sync with HQ server at strategic timings of the day

Customer website:

* Catalog
  + View & search product catalog with pagination
  + Add/remove product to shopping cart
  + View total price of products in shopping cart
  + Checkout & confirm shopping cart
  + Specify delivery address for online order
* Online orders
  + View, search & sort list of online orders from the customer
  + View detailed information for each order
* Store locator
  + View all available outlets
  + Locate outlets on Google maps
  + Find direction route from address specified to outlet selected
* Account management
  + View account information
  + Edit account address and phone number
  + Facebook login/logout

Implementation

Database design:   
  
Shop database

**Open API:**

1. Facebook:
   1. OAuth
   2. Comments plugin
2. Google maps:
   1. Geolocator
   2. Direction service

**Software framework/libraries:**

1. MySQL
2. Node.js
3. jQuery
4. jQuery UI
5. Block UI
6. Editable Grid
7. Highcharts
8. Twitter bootstrap – carousel, modal, dropdown, alerts

**Implementation:**

Hyper Market Management system has been designed to accommodate the assumed workflow :

1. **Sync at start of the day –** As per the functioning of the Hyper Market System, it has been assumed that a regular day starts of with an initial sync of the inventory of the outlet. There are two reasons for doing so at the start of the day:
   1. **Updated Stock:** Stock requests of an outlet are pushed to the HQ at the end of the day, and the stock order placed can arrive in the interim between the end of the working day and the start of the next day. Hypermarket Management System assumes this interim to be between 11 PM of the previous day and 6:30 AM the next day.
   2. **Recomputed Prices:** The dynamic pricing model of the Hypermarket Management System is designed to do the computation of the prices of the next day *before* the start of the next day. The mathematical model relies on factors such as sales made the previous day, the current stock and last day’s prices.

For scheduling the above sync, the Hypermarket Management System uses Cron, a Node.JS module. This module provides an API for scheduling tasks and has been therefore been used for the automation of the syncing operations of the Hyper Market Management System.

1. **Transactions –** A MySQL driver for Node.JS is used for processing transactions and restock requests of different products, as these operations as carried out through constant querying of the outlet database**.**On the HQ side, the following operations can be carried out:
   1. Addition of new product
   2. Editing a new product
   3. Discontinuation of a product
   4. Addition of an Outlet, and so on

Changes made in the following are reflected immediately in the Online Shopping and at the end of the day for a regular outlet.

1. **Sync at end of the day –** Hypermarket Management System carries out a final sync at the end of the day which consists of the following:
   1. **Inventory Pull from HQ –** In this operation, the Outlet system pulls updated inventory data from HQ to update changes such as adding new products to inventory, discontinuation of products etc.
   2. **Push of Transaction Data –** Records of all the transactions made during the day are pushed to the server for logging and generating outlet statistics.
   3. **Push of Restock Requests –** Finally, all restock requests generated during the day as a result of transactions are pushed to HQ to be processed.

In order to deal with the huge volume of data which is being constantly pulled from and pushed to the server during the sync operations, a specialized mechanism is being used to ensure efficient communication.

**Push :** The data to be pushed to the HQ server is divided into segments/packets of 2000 elements, which are then transmitted one by one.

**Pull :** The process of pulling data from the HQ is carried out in two steps :

1. A *probe* request is first made by the outlet server to determine the size of the data to be retrieved from the server
2. On the basis of the size of data to be retrieved from HQ, and the maximum segment size, a suitable number of requests are made, with each segment being retrieved per call.

The above mechanism utilizes Async.JS and request.js modules to implement the abovementioned functionality.

**Special Features :**

**Email Notifications :** For all transactions that are carried out through the online shopping portal, users receive a confirmation email notifying them of their purchase. Furthermore, once an order has been dispatched from the HQ, the user gets another mail informing him of the status.  
Email.JS, another Node.JS module is used for this purpose.

**Statistics :** On the basis of the transaction data retrieved from the different outlets, a pie chart and a column chart are generated. The pie chart indicates the fraction of the total revenue generated by each of the outlets, whereas the line graph is outlet specific, depicting the best selling products in the last week along with the revenue earned generated by them. The statistics rely on Highcharts.js for generating the said charts.

Test

Setup:

*Datasets:*   
Ultimate size 10000

*Hosting:*   
HQ server, customer website on AWS EC2  
Shop PC on localhost

*Network:*   
NUS wireless network

Test cases:

**[HQ] Add new product**

Success:   
Barcode – positive 8 digit integer  
Product name – string > 150 characters  
Category – string > 100 characters  
Manufacturer – positive 4 digit integer  
Cost price – positive float  
Image URL – optional, no constraints

Failure:   
Empty fields  
Wrong data type  
Barcode not unique  
Input length

**[HQ] Add product to outlet inventory**

Success:   
Outlet – item selected  
Selling price – positive float  
Minimum stock – positive integer

Failure:   
Empty fields  
Wrong data type  
No outlet selected

**[HQ] Add new outlet**

Success:   
Outlet name – string > 150 characters  
Address – string > 100 characters  
Longitude - optional  
Latitude - optional

Failure:   
Empty field  
Exceed maximum length  
No existing address on Google maps

**[Shop PC] Add new transaction**

Success:   
Quantity – positive integer, not more than stock  
Barcode – corresponds to existing barcode

Failure:   
Empty transaction  
Negative quantity  
Quantity exceed available stock  
Wrong datatype of quantity  
No existing product barcode in outlet inventory

**[Shop PC] Add (/Edit) new price display**

Success:   
Display ID – integer  
Barcode – integer, correspond to existing barcode  
Description – string, <40 characters

Failure:   
Empty fields   
Wrong datatypes  
Display ID already exists  
No existing product barcode in outlet inventory

**[HQ] Discontinue product**

Handling exceptions:   
Product does not exist in outlet yet  
Product has been discontinued before

**[Customer website] Checkout catalog**

Success:   
Quantity - positive, not exceeding available stock

Failure:  
Empty shopping cart  
Negative quantity  
Quantity exceed available stock  
Wrong datatype of quantity