**Project team 18- Retail Warehouse Management**

**Team Members:**

**Chinmay Rawool**

**Nitish Rangarajan**

**Raka Choudhury**

**Vimoka Nalluri**

**PROJECT PROPOSAL**

**Overview**:

The project aims to implement a real-time retail warehouse management system which manages the overall functioning of a retail warehouse. Goods have to be supplied to the store upon request and warehouses store goods that were sent by supplier. This application would keep an entry of every good in the warehouse in databases and track its availability.

**Scope and Objectives:**

**The application will track the various resources available in a warehouse.**Users can create purchase orders containing the required items, supplier# and warehouse#. This purchase order is then sent to the supplier.. Supplier processes the order and sends the required items to the warehouse. Warehouse checks for surplus or deficit items in the carton. The items are stored in the warehouse databases. Stores can lookup into these databases and request for items from the warehouse on demand.

Modules:

1. Creating a warehouse database containing items currently in the warehouse using the item\_description, warehouse# and the supplier# from which it was received as its attributes.
2. Creating a purchase order to be sent to the supplier.
3. Track the availability of items.
4. Store looks up into the warehouse’s database.

**Project Environment**

We will be using NetBeans which is an open-source IDE(Integrated Development environment). We will be implementing the project using JSP(JavaServer Pages). The project follows a Client-Server model. We will be using servlets to bind MySQL database and JSP. We would also be using JDBC API. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. GIT and GITHUB is used to collaborate with team member. We would be implementing the project on Windows OS machine.

**High Level Requirements**

**Initial User Roles**

1. **Stores Representative**

The user can be the store manager or a person who works in a particular store and takes care of requesting items from the warehouse on behalf of the store depending on the requirements. They can place the order on the application which contains Item description and Quantity of items needed.

2. **Warehouse Manager**

The person scans the items mentioned in the order placed by stores representative.If the goods are available,he sends them out for delivery and updates the order status.If the goods are deficit in number he sends a request to POM User to order for the remaining items from the appropriate store. If the goods received are surplus in number, the Warehouse user can decide whether to keep the items in the warehouse or return it back to the supplier.

3. **POM (Purchase Order Management) User**

The user responsible for creating purchase orders which contains the goods required and the warehouse location where items are to be sent. It is sent to the supplier in an EDI format which includes Purchase Order#, Delivery Instruction, Item Description, sku(size) details and the warehouse location.

4. **Supplier**

The person who validates the purchase orders created by POM User and sends the required items to the warehouse location mentioned in the EDI.

**Initial User Stories**

1. As a store representative,I want to place an order to the warehouse and receive ordered items so that I can get things as per my need.
2. As a warehouse manager,I want to login and logout of the Retail Warehouse Management system so that I can manage the orders received from the supplier.
3. As a warehouse manager,I want to view a received order from the store and check if ordered items are available in the warehouse.
4. As a warehouse manager,I want to view list of available delivery person and assign delivery duties to them so that the store representative can get the items he ordered for.
5. As a warehouse manager,I want to place a request for the required items to the POM User so that the warehouse can be refilled.
6. As a POM User,I want to place an order to the appropriate supplier.
7. As a supplier,I want to view the orders received from POM User and send the required items.
8. As a warehouse manager, I want to check the receipts from the supplier so that I can check for deficit/surplus items.
9. As a warehouse manager, I want to report the deficit items to the supplier so that the supplier can ship the missing items.
10. As a warehouse manager, I want to inform the supplier about the surplus items so that the warehouse can store the additional items.
11. As a store representative,I want to view and update an order that I already placed.
12. As a warehouse manager,I want to change the status of a Purchase order that was logged by POM User.
13. As a store representative,I want to track an order that I already placed so that I have an idea when the expected delivery is.

**High Level Conceptual Design**

Some of the expected main entities are as follows :

Store\_representative

Warehouse\_manager

POM\_user

Supplier

Store\_order

Supplier\_order

Warehouse\_stock

POM\_request

Delivery\_person

Delivery\_schedule

Return\_request

Some of the possible main relationships are as follows :

1. Store\_representative creates Store\_order
2. Store\_representative creates Return\_request
3. Warehouse\_manager manages Warehouse\_stock for a Warehouse
4. Warehouse\_manager creates POM\_request
5. Warehouse\_manager creates Delivery\_schedule
6. Warehouse\_manager updates Store\_Order
7. POM\_user creates Supplier\_order
8. Warehouse\_manager updates Supplier\_order

SPRINT 1

**User stories chosen for inclusion in Sprint 1**

1. **As a store representative,I want to place an order to the warehouse and receive ordered items so that I can get things as per my need.**

Notes:

i) User should be logged in as store representative to be able to place order.Add login/logout user stories.

ii) A store representative should add one item at a time in an order. The story should reflect that.

iii) A list of items to choose from should be available to the store representative.

iv) An order that is placed should have at least 1 item added from the itemlist.

Refined user stories:

a) As a store representative,I want to be able to login to the retail warehouse management system.

b) As a store representative,I want to be able to logout from the retail warehouse management system.

c) As a store representative,I want to view list of available items.

d) As a store representative,I want to be able to perform basic search for items on the list of available items.

e) As a store representative,I want to be able to add an item to my cart.

f) As a store representative,I want to be able to remove an item from my cart.

g) As a store representative,I want to be able to place an order to the warehouse if my cart has at least 1 item.

h) As a store representative,I want to be able to select a preferred time slot for delivery of my order.

**2.** **As a warehouse manager,I want to login and logout of the Retail Warehouse Management system so that I can manage the orders received from the store.**

Notes:

i) User should be able to login as a warehouse manager. A warehouse manager should be able to log out of the system when needed. Separate the login-logout features into different user stories.

Refined user stories:

a) As a warehouse manager, I want to be able to login to the retail warehouse management system.

b) As a warehouse manager, I want to be able to logout from the retail warehouse management system.

**3. As a warehouse manager, I want to view a received order from the store and check if ordered items are available in the warehouse.**

Notes:

i) A warehouse manager should be able to view the received orders.

ii) A warehouse manager should be able to view the items available in stock in the warehouse.

iii) A warehouse manager should be able to check the availability of each item as per request.

Refined user stories:

a) As a warehouse manager, I want to view list of orders received from store representatives.

b) As a warehouse manager, I want to view list of items requested in each order.

c) As a warehouse manager, I want to check the availability of the items as requested.

d) As a warehouse manager, I want the requested items that are in stock to be marked as ‘Available for delivery’.

e) As a warehouse manager, I want the requested items that are not in stock to be marked as ‘Not in Stock’.

**4. As a store representative,I want to view and update an order that I already placed.**

Notes:

**i)** Store representative will be able to check their existing orders to the warehouse.

ii) The representative can update the order if more or less items are needed so that he can increase or decrease the number of items in the order.

iii) The representative can cancel the order if the items exist in the store.

Refined user Stories:

a) As a Store representative, I want to look at my existing orders upon logging into the system.

b) As a Store representative, I want to increase the number of items in the order so that I can meet the demands of the store.

c) As a Store representative, I want to decrease the number of items in the order so that I can meet the demands of the store.

d) As a Store representative, I want to cancel the items in the order so that I can meet the demands of the store.

**List of refined user stories considered for Sprint-1(in decreasing order of priority)**

1. As a store representative,I want to be able to login to the retail warehouse management system.
2. As a store representative,I want to view list of available items.
3. As a store representative,I want to be able to add an item to my cart.
4. As a store representative,I want to be able to place an order to the warehouse if my cart has at least 1 item.
5. As a store representative,I want to be able to select a preferred time slot for delivery of my order.
6. As a warehouse manager, I want to be able to login to the retail warehouse management system.
7. As a warehouse manager, I want to view list of orders received from store representatives.
8. As a warehouse manager, I want to view list of items requested in each order.
9. As a warehouse manager, I want to check the availability of the items as requested.
10. As a warehouse manager, I want the requested items that are not in stock to be marked as ‘Not in Stock’.
11. As a Store representative, I want to look at my existing orders upon logging into the system.
12. As a store representative,I want to be able to logout from the retail warehouse management system.
13. As a warehouse manager, I want to be able to logout from the retail warehouse management system.

**Sprint-1**

**Part 1** : **Conceptual Design**

List of entities considered in this sprint

1. **Entity:** Store\_representative

**Attributes:**

store\_id

user\_id

1. **Entity:** Warehouse\_manager

**Attributes:**

warehouse\_id

user\_id

1. **Entity:**Store\_order

**Attributes:**

order\_id

item\_id

quantity\_ordered

store\_id

delivery\_date

warehouse\_id

Status

1. **Entity:** Warehouse\_stock

**Attributes:**

warehouse\_id

item\_id

quantity\_available

Relationship : **Store\_representative** creates **Store\_order**

**Cardinality:** many to many

**Participation:**

Store representative has partial participation

Store order has total participation

Relationship : **Warehouse\_manager** manages **Warehouse\_stock** for a **Warehouse**

**Cardinality:** many to one

**Participation:**

Warehouse\_manager has total participation

Warehouse\_stock has total participation

**Part 2 : Logical Design**

**Table**: user

**Columns:**

username

password

first\_name

last\_name

uid [ Primary key ]

designation

**Table :**  store\_representative

**Columns:**

store\_id [ Foreign key; references id of the store table ]

uid [ Primary key, Foreign key; References Id of User table]

Justification: Using foreign key approach to represent the “specialization of the User table”

**Table :**  warehouse\_manager

**Columns:**

warehouse\_id [ Foreign key; referencing id of the warehouse table ]

uid [ Primary key, Foreign key; References Id of User table]

Justification: Using foreign key approach to represent the “specialization of the User table”

**Table :**  warehouse

**Columns:**

warehouse\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

state

city

zipcode

**Table :**  store

**Columns:**

store\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

state

city

zipcode

warehouse\_id - [References to primary key of Table Warehouse]

Justification: For the relationship Store connects to Warehouse, it is a many to one relationship.

We use foreign key approach to represent this in database.

**Table:** store\_order

**Columns:**

order\_id [ Primary Key ]

store\_id [ Foreign key; references id of the store table ]

delivery\_date

total\_price

warehouse\_id [ Foreign key; references id of the warehouse table ]

status

Justification: For the relationship store representative creates store order for a store, it is one to many relationship. We use foreign key approach to represent this in the database.

**Table:** store\_order\_item

order\_id [ Foreign key; references id of the store\_order table ]

item\_id [ Foreign key; references id of the global\_item table ]

quantity\_ordered

Justification: For the relationship store\_order has store\_order\_item, it is one to many relationship. We use foreign key approach to represent this in the database. This table doesn’t contain primary key as this is dependent on store\_order table.

**Table:** global\_item

**Columns:**

item\_id [ Primary key ]

name

description

price\_per\_unit($)

**Table :** warehouse\_stock

**Columns:**

warehouse\_id [ Primary key,Foreign key; references id of the warehouse table ]

item\_id [Primary key, Foreign key; references item\_id of the Global\_item table ]

quantity\_available

[ Primary key ] - warehouse\_id plus the item\_id can together be the primary key for the warehouse\_stock table

Justification : The relationship Warehouse Manager manages warehouse\_stock, it is a many to one relationship. We use the warehouse\_id and item\_id as the foreign key approach to represent it in the database.

**Part 3 : Implementation**

SQL Dump for database : retail1 has been included with the submission.

**Part 4 : User interface**

The user interface has been created using JSP and servlets. Depending on different users, the pages are directed to different Servlet which takes care of the HTTP request and respond.

SPRINT 2

**List of User stories considered for Sprint 2 :**

1. As a store representative,I want to be able to perform basic search for items I need.
2. As a store representative,I want to be able to remove an item from my cart.
3. As a Store representative, I want to update the number of items in an already placed active order so that I can meet the demands of the store.
4. As a Store representative, I want to cancel an already placed active order.
5. **As a warehouse manager, I want to place a request for the required items to the POM User so that the warehouse can be refilled.**

Notes:

i) A warehouse manager should be able to request POM User to generate new order for the supplier for all the deficit items.

ii) A warehouse manager should be able to add details about a Store\_order for each requested item. Details like which store ordered for the item and when the order was placed.

Refined user stories:

a) As a warehouse manager, I want to identify list of all required items that are Not in Stock in the warehouse currently.

b) As a warehouse manager, I want to generate a request for the POM User for each item that is Not in Stock.

c) As a warehouse manager, I want to associate with each POM Request all relevant information about a store order for which a particular item is being requested so that POM User knows the order corresponding to requested items.

**List of refined user stories considered for Sprint-2(in decreasing order of priority)**

1. As a store representative,I want to be able to remove an item from my cart.
2. As a Store representative, I want to update the number of items in an already placed active order so that I can meet the demands of the store.
3. As a Store representative, I want to cancel an already placed active order.
4. As a warehouse manager, I want to identify list of all required items that are Not in Stock in the warehouse currently.
5. As a warehouse manager, I want to generate a request for the POM User for each item that is Not in Stock.
6. As a POM User,I want to create a Purchase Order for items that warehouse manager requested for.

**Part 1 : Conceptual Design**

List of entities considered in this sprint :

1. **Entity:** store\_representative

**Attributes:**

store\_id

uid

2. **Entity:** store\_order

**Attributes:**

delivery\_date

order\_id

status

store\_id

total\_price

Warehouse\_id

3. **Entity:** store\_order\_item

**Attributes:**

item\_id

order\_id

quantity\_ordered

4. **Entity:** warehouse

**Attributes:**

warehouse\_id

name

address\_line\_1

address\_line\_2

state

city

Zipcode

5. **Entity:** warehouse\_manager

**Attributes:**

uid

Warehouse\_id

6. **Entity:** warehouse\_stock

**Attributes:**

item\_id

warehouse\_id

quantity\_available

7. **Entity:** warehouse\_order

**Attributes:**

item\_id

order\_id

quantity\_required

8. **Entity:** purchase\_order

**Attributes:**

delivery\_date

order\_id

po\_id

po\_status

supplier\_id

Warehouse\_id

9. **Entity:** purchase\_order\_item

**Attributes:**

item\_id

order\_id

quantity\_ordered

Relationship : **Store\_representative** creates **Store\_order**

**Cardinality:** many to many

**Participation:**

Store representative has partial participation

Store order has total participation

**Relationship :** **Store\_representative** updates items in **Store\_order\_item**

**Cardinality:** many to many

**Participation:**

Store representative has partial participation

Store order item has total participation

**Relationship : Store\_representative** cancels **Store\_order**

**Cardinality:** many to many

**Participation:**

Store representative has partial participation

Store order has total participation

**Relationship : Warehouse\_manager** manages **Warehouse\_stock**

**Cardinality:** many to one

**Participation:**

warehouse\_manager has total participation

warehouse\_stock has total participation

**Relationship :** **Warehouse\_manager** creates **Warehouse\_order**

**Cardinality:** many to one

**Participation:**

Warehouse\_manager has total participation

Warehouse\_order has total participation

**Relationship:** **Warehouse\_manager** requests **pom user**

**Cardinality:** many to one

**Participation:**

warehouse\_manager has partial participation

Pom\_user has total participation

**Relationship:** **pom user** creates **purchase order**

**Cardinality:** one to many

**Participation:**

Pom\_user has partial participation

Purchase\_order has total participation

**Part 2: Logical Design**

**Table**: user

**Columns:**

username

password

first\_name

last\_name

uid [ Primary key ]

designation

**Table :**  store

**Columns:**

store\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

Zipcode

warehouse\_id - [References to primary key of Table Warehouse]

Justification: For the relationship Store connects to Warehouse, it is a many to one relationship.

We use foreign key approach to represent this in database.

**Table:** store\_representative

**Columns:**

store\_id [ Foreign key; references id of the store table ]

uid [ Primary key ]

**Justification:** using foreign key approach to represent the many to many relationship of store having a store representative

**Table:** store\_order

**Columns:**

delivery\_date

order\_id [ Primary key ]

status

store\_id [ Foreign key; references id of the store table ]

total\_price

**Justification:** For the relationship store representative creates store order for a store, it is one to many relationship. We use foreign key approach to represent this in the database.

**Table:** store\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the global item table ]

order\_id [ Foreign key; references id of the store order table ]

quantity\_ordered

**Justification:**  For the relationship store\_order has store\_order\_item, it is one to many relationship. We use foreign key approach to represent this in the database. This table doesn’t contain primary key as this is dependent on store\_order table.

**Table:** warehouse

**Columns:**

warehouse\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

zipcode

**Table:** warehouse\_manager

**Columns:**

uid [ Primary key ]

warehouse\_id [ Foreign key; references id of the warehouse table ]

**Justification:** using foreign key approach to represent the many to many relationship of warehouse having a warehouse manager

**Table:** warehouse\_stock

**Columns:**

item\_id [ Primary key ]

warehouse\_id [ Primary key ]

quantity\_available

**Table:** warehouse\_order

**Columns:**

item\_id [ Foreign key; references id of the warehouse stock table ]

order\_id [ Foreign key; references id of the store order table ]

quantity\_ordered

**Justification :** for the relationship where the warehouse manager creates list of unavailable items in the warehouse is a one to many relationship.we use foreign key approach to represent the same. This table doesnt have a primary key as it is dependent on the warehouse\_stock table.

**Table:** global\_item

**Columns:**

item\_id [ Primary key ]

name

description

price\_per\_unit($)

**Table:** location

**Columns:**

zipcode [ Primary key ]

state

City

**Table:** pom\_userlist

**Columns:**

**uid** [ Primary key ]

warehouse\_id [ Primary key ]

**Table:** purchase\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the warehouse stock table ]

po\_id [ Foreign key; references id of the store order table ]

quantity\_ordered

**Justification:** For the relationship Pom user creates purchase order for the warehouse, it is a one to many relationship. We use foreign key approach to represent this in the database.

**Table:** purchase\_order

**Columns:**

order\_id [ [ Foreign key; references id of the store order table ]

po\_id [ Primary key ]

po\_status

supplier\_id [ Foreign key; references id of the supplier table ]

**Justification:** using foreign key approach to represent the one to many relationship of the pom requesting items to the supplier

**Table:** supplier

**Columns:** supplier\_id {primary key]

supplier\_name

address\_line\_1

address\_line\_2

zipcode

**Table:** supplier\_stock

**Columns:**

item\_id [ Primary key ]

supplier\_id [ Primary key ]

quantity\_available

**Part 3 - Normalization**

1. **Table**: user

**Columns:**

username

password

first\_name

last\_name

uid [ Primary key ]

Designation

**Highest Normalization Level** : <4NF>

**2. Table :**  store

**Columns:**

store\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

Zipcode

warehouse\_id - [References to primary key of Table Warehouse]

**Highest Normalization Level** : <4NF>

**3. Table:** store\_representative

**Columns:**

store\_id [ Foreign key; references id of the store table ]

uid [ Primary key ]

**Highest Normalization Level** : <4NF>

**4. Table:** store\_order

**Columns:**

delivery\_date

order\_id [ Primary key ]

status

store\_id [ Foreign key; references id of the store table ]

Total\_price

**Highest Normalization Level** : <4NF>

**5. Table:** store\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the global item table ]

order\_id [ Foreign key; references id of the store order table ]

Quantity\_ordered

**Highest Normalization Level** : <4NF>

**6. Table:** warehouse

**Columns:**

Warehouse\_id [ Primary key ]

name

address\_line\_1

Address\_line\_2

zipcode [ Foreign key; references zipcode of the location table ]

**Highest Normalization Level** : <4NF>

Justification : creating a new table location with zipcode as the primary key to resolve the transitive dependency

**7. Table:** Location

**Columns:**

Zipcode [ Primary key ]

state

city

**Highest Normalization Level** : <4NF>

**8. Table:** warehouse\_manager

**Columns:**

uid [ Primary key ]

warehouse\_id [ Foreign key; references id of the warehouse table ]

**Highest Normalization Level** : <4NF>

**9. Table:** warehouse\_stock

**Columns:**

item\_id [ Primary key ]

warehouse\_id [ Primary key ]

quantity\_available

**Highest Normalization Level** : <4NF>

10. **Table:** warehouse\_order

**Columns:**

item\_id [ Foreign key; references id of the warehouse stock table ]

order\_id [ Foreign key; references id of the store order table ]

quantity\_required

required\_delivery\_date

**Highest Normalization Level** : <4NF>

**11. Table:** global\_item

**Columns:**

item\_id [ Primary key ]

name

description

price\_per\_unit($)

**Highest Normalization Level** : <4NF>

**12. Table:** pom\_userlist

**Columns:**

**uid** [ Primary key ]

warehouse\_id [ Primary key ]

**Highest Normalization Level** : <4NF>

**13. Table:** purchase\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the warehouse stock table ]

order\_id [ Foreign key; references id of the store order table ]

quantity\_ordered

**Highest Normalization Level** : <4NF>

**14. Table:** purchase\_order

**Columns:**

order\_id [ [ Foreign key; references id of the store order table ]

po\_id [ Primary key ]

po\_status

supplier\_id [ Foreign key; references id of the supplier table ]

**Highest Normalization Level** : <4NF>

**15. Table:** supplier

**Columns:** supplier\_id {primary key]

supplier\_name

address\_line\_1

address\_line\_2

Zipcode

**Highest Normalization Level** : <4NF>

**16. Table:** supplier\_stock

**Columns:**

item\_id [ Primary key ]

supplier\_id [ Primary key ]

quantity\_available

**Highest Normalization Level** : <4NF>

**Part 4 - Implementation**

SQ L Dump for database : retail1 has been included with the submission.

***Sprint 3***

**List of refined stories for Sprint 3**

1. As a supplier,I want to view the orders received from POM User and send the required items.

**Notes:**

1. Supplier can view the requests from the POM user.
2. Supplier responds to the request by sending the requested items to the warehouse.

**Refined Stories:**

1. As a supplier,I want to view the orders received from POM User.
2. As a supplier,I want to send the requested items to the warehouse.
3. As a warehouse manager, I want to check the receipts from the supplier so that I can check for deficit/surplus items.

**Notes**:

1. Warehouse manager checks for the receiver receipts i.e the items mentioned in the carton.
2. Warehouse manager checks for the stock ledger receipts i.e scanning the items individually after opening the carton.

**Refined Stories:**

1. As a warehouse manager, I want to check for the items mentioned in the carton label.
2. As a warehouse manager, I want to check for all the items individually after opening the carton.
3. As a warehouse manager, I want to report the deficit items to the supplier so that the supplier can ship the missing items.

**Notes:**

1. Warehouse reports the missing items to the supplier.

**Refined Stories:**

1. As a warehouse manager, I want to report the deficit items to the supplier so that the supplier can ship the missing items.
2. As a warehouse manager, I want to inform the supplier about the surplus items so that the warehouse can store the additional items.

**Notes:**

1. Warehouse reports the surplus items to the supplier.

**Refined Stories:**

1. As a warehouse manager, I want to report the surplus items to the supplier
2. As a store representative,I want to view and update an order that I already placed.

**Notes**:

1. Store representative can update the orders by increasing the quantities in the order.

**Refined Stories:**

1. As a store representative,I want to update the orders by increasing/decreasing the quantities in the order.
2. As a warehouse manager,I want to change the status of a Purchase order that was logged by POM User.

**Notes:**

1. Warehouse manager changes the status of the Purchase order to ‘Complete’ once the items are received.
2. As a store representative,I want to track an order that I already placed so that I have an idea when the expected delivery is.

**Notes:**

1. Store representative can track the orders.

**Refined User Stories (decreasing order of priority):**

1. As a supplier,I want to view the orders received from POM User.
2. As a supplier,I want to send the requested items to the warehouse.
3. As a warehouse manager, I want to check for the items mentioned in the carton label.
4. As a warehouse manager, I want to check for all the items individually after opening the carton.
5. As a warehouse manager, I want to report the deficit items to the supplier so that the supplier can ship the missing items.
6. As a warehouse manager, I want to report the surplus items to the supplier
7. As a store representative,I want to update the orders by increasing/decreasing the quantities in the order.
8. As a warehouse manager,I want to change the status of a Purchase order that was logged by POM User.
9. As a store representative,I want to track an order that I already placed so that I have an idea when the expected delivery is.

**Sprint 3 -Final Submission**

**Part 1 : Conceptual Design**

List of entities considered in this sprint :

1. **Entity:** User

**Attributes:**

Username

Password

First\_name

Last\_name

uid

Designation

Store\_representative is a specialization of User

Supplier\_representative is a specialization of User

POM(Purchase Order Manager) is a specialization of User

Warehouse Manager is a specialization of User

2. **Entity :**  Store

**Attributes:**

store\_id

name

address\_line\_1

Address\_line\_2

city

state

Zipcode

**3. Entity:** Store\_order

**Attributes:**

delivery\_date

order\_id

status

Total\_price

**4. Entity :** Warehouse

**Attributes :**

Warehouse\_id

Name

Address\_line\_1

Address\_line\_2

City

State

Zipcode

**5. Entity :** Global\_items

**Attributes :**

Item\_id

Name

Description

Price\_per\_unit($)

6. **Entity :** Supplier

**Attributes :**

supplier\_id

supplier\_name

address\_line\_1

Address\_line\_2

City

State

Zipcode

7. **Entity :** Carton

**Attributes:** Carton\_id

Item

8. **Entity:** Report

**Attributes:**

Carton\_id

Item\_id

Quantity

Report\_type

9. **Entity:** Purchase\_order

**Attributes:**

Po\_id

Po\_status

Item

**Relationships**

Relationship: **Store** contacts **Warehouse**

Cardinality : many to one

Participation : Store has total participation

Warehouse has partial participation

Relationship: **Store\_representative** creates **Store\_Order**

Cardinality : one to many

Participation : Store has partial participation

Store\_Order has total participation

Relationship : **store\_order** contains **global\_items**

Cardinality:many to many

Participation:

**store\_order** has total participation

**global\_items** has partial participation

Relationship : **supplier\_representative** creates **supplier\_dispatch\_carton**

Cardinality : one to many

Participation : supplier\_representative has partial participation

supplier\_dispatch\_carton has total participation

Relationship : **warehouse\_manager** creates **report** to the **supplier\_representative**

Cardinality : many to many

Participation : warehouse\_manager has partial participation

Report has total participation

Relationship : **carton** contains **global\_items**

Cardinality:many to many

Participation:

global\_items has partial participation

carton has partial participation

Relationship : **supplier\_representative** sends **carton**

Cardinality:one to many

Participation:

Supplier\_representative has partial participation

carton has total participation

Relationship : **warehouse** contains **global\_items**

Cardinality:many to many

Participation:

**warehouse** has partial participation

**global\_items** has partial participation

Relationship : **purchase\_order** contains **global\_items**

Cardinality:many to many

Participation:

**purchase\_order** has total participation

**global\_items** has partial participation

Relationship : **supplier** contains **global\_items**

Cardinality:many to many

Participation:

**supplier** has partial participation

**global\_items** has partial participation

Action : **Store\_representative** adds items in **Store\_order**

Action : **Store\_representative** deletes items in **Store\_order**

Action : **POM** adds items in **purchase\_order**

Action : **POM** deletes items from **purchase\_order**

Action : **warehouse\_manager** updates warehouse\_stock

**Part 2: Logical Design,Normalization,Indexes**

**1. Table**: user

**Columns:**

Username

password

first\_name

last\_name

uid [ Primary key ]

Designation

**Highest Normalization Level** : <3NF>

Justification: username is unique and is one of the candidate key. Violates Boyce-Codd Normal Form.

**Indexes:**

**Index 1:** Clustered- uid

Justification: uid is the primary key and is used to refer the user table as it identifies each row uniquely.

**Index 2:** Non clustered- username

Justification: username can be used for validation and login. So providing indexing for username will reduce sorting time.

**Index 3:** Non Clustered- firstname, lastname

Justification: This index helps in displaying or searching for user as per his/her name.

**Index 4:** Non Clustered- designation

Justification: To view all the users as per a particular specialization.

**2. Table :**  store

**Columns:**

store\_id [ Primary key ]

name

address\_line\_1

address\_line\_2

Zipcode [ Foreign key; references id of the zipcode table]

warehouse\_id [Foreign key;References to primary key of Table Warehouse]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- store\_id

Justification: store\_id is the primary key and is used to refer the store table as it identifies each row uniquely.

**Index 2:** Non clustered- zipcode

Justification: indexing on zip code will help to display all the store with same zip code.

**3. Table:** store\_representative

**Columns:**

store\_id [ Foreign key; references id of the store table ]

uid [ Primary key,Foreign key; references id of the user table ]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- uid

Justification: uid is the primary key and is used to refer the store\_representative table as it identifies each row uniquely.

**Index 2:** Non clustered- store\_id

Justification: indexing on store\_id will help to display all the users/store\_representatives of a particular store.

**4. Table:** store\_order

**Columns:**

delivery\_date

order\_id [ Primary key ]

status

store\_id [ Foreign key; references id of the store table ]

Total\_price

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- order\_id

Justification:order\_id is the primary key and is used to refer the store\_order table as it identifies each row uniquely.

**Index 2:** Non clustered- store\_id

Justification: indexing on store\_id will help to display all the orders created by a particular store.

**Index 3:** Non clustered- status

Justification: indexing on status will help to display all the orders under a particular status say “Order Placed”.

**5. Table:** store\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the global item table ]

order\_id [ Foreign key; references id of the store order table ]

Quantity\_ordered

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Non clustered- order\_id

Justification: indexing on order\_id will help to display all the items in a particular order.

**Index 2:** Non clustered- item\_id

Justification: indexing on item\_id will help to display all the orders that include a particular item or the number of quantities that have been ordered for a particular item.

**6. Table:** warehouse

**Columns:**

Warehouse\_id [ Primary key ]

name

address\_line\_1

Address\_line\_2

zipcode [ Foreign key; references zipcode of the location table ]

**Highest Normalization Level** : <4NF>

Justification : creating a new table location with zipcode as the primary key to resolve the transitive dependency.

**Indexes:**

**Index 1:** Clustered- warehouse\_id

Justification: warehouse\_id is the primary key and is used to refer the warehouse table as it identifies each row uniquely.

**Index 2:** Non clustered- zipcode

Justification: indexing on zip code will help to display all the store with same zip code.

**7. Table:** Location

**Columns:**

zipcode [ Primary key ]

state

city

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- zipcode

Justification: zipcode is the primary key and is used to refer the Location table as it identifies each row uniquely.

**Index 2:** Non**-**Clustered- state

Justification: indexing on state will help to display all the warehouses or stores or suppliers located in a particular state.

**Index 2:** Non**-**Clustered- city

Justification: indexing on city will help to display all the stores or suppliers located in a particular city.

**8. Table:** warehouse\_manager

**Columns:**

uid [ Primary key,Foreign key; references id of the user table ]

warehouse\_id [ Foreign key; references id of the warehouse table ]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- uid

Justification: uid is the primary key and is used to refer the warehouse\_manager table as it identifies each row uniquely.

**Index 2:** Non clustered- warehouse\_id

Justification: indexing on warehouse\_id will help to display all the users/warehouse\_managers of a particular warehouse.

**9. Table:** warehouse\_stock

**Columns:**

item\_id [ Primary key,Foreign key; references id of the global\_item table ]

warehouse\_id [ Primary key,Foreign key; references id of the warehouse table ]

quantity\_available

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- item\_id,warehouse\_id

Justification:{item\_id,warehouse\_id} is the primary key and is used to refer the warehouse\_stock table as it identifies each row uniquely.

10. **Table:** warehouse\_order

**Columns:**

item\_id [ Primary key,Foreign key; references id of the warehouse stock table ]

order\_id [Primary key, Foreign key; references id of the store order table ]

quantity\_required

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- item\_id,order\_id

Justification:{item\_id,order\_id} is the primary key and is used to refer the warehouse\_order table as it identifies each row uniquely.

**11. Table:** global\_item

**Columns:**

item\_id [ Primary key ]

name

description

price\_per\_unit($)

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- item\_id

Justification:item\_id is the primary key and is used to refer the global\_item table as it identifies each row uniquely.

**12. Table:** pom\_userlist

**Columns:**

uid [ Primary key,Foreign key: References to primary key of Table user ]

warehouse\_id [Primary key, Foreign key: References to primary key of Table Warehouse]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- uid

Justification:uid is the primary key and is used to refer the pom\_userlist

table as it identifies each row uniquely.

**Index 2:** Non clustered- warehouse\_id

Justification: indexing on warehouse\_id will help to display all the users/pom\_users of a particular warehouse.

**13. Table:** purchase\_order\_item

**Columns:**

item\_id [ Foreign key; references id of the global\_item table ]

po\_id [ Foreign key; references id of the store order table ]

quantity\_ordered

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Non clustered- po\_id

Justification: indexing on po\_id will help to display all the items in a particular purchase order.

**Index 2:** Non clustered- item\_id

Justification: indexing on item\_id will help to display all the purchase\_orders that include a particular item or the number of quantities that have been ordered for a particular item.

**14. Table:** purchase\_order

**Columns:**

order\_id [ [ Foreign key; references id of the store order table ]

po\_id [ Primary key ]

po\_status

supplier\_id [ Foreign key; references id of the supplier table ]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- po\_id

Justification:po\_id is the primary key and is used to refer the purchase\_order

table as it identifies each row uniquely.

**Index 2:** Non clustered- supplier\_id

Justification: indexing on supplier\_id will help to display all the purchase\_orders created for a particular supplier.

**Index 3:** Non clustered- po\_status

Justification: indexing on po\_status will help to display all the purchase\_orders that have a particular status like “placed”.

**15. Table:** supplier

**Columns:** supplier\_id [primary key]

supplier\_name

address\_line\_1

address\_line\_2

Zipcode [ Foreign key; references id of the zipcode table]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- supplier\_id

Justification:supplier\_id is the primary key and is used to refer the supplier

table as it identifies each row uniquely.

**Index 2:** Non clustered- zipcode

Justification: indexing on zip code will help to display all the suppliers with same zip code.

**16. Table:**supplier\_dispatch\_carton

**Columns:**

carton\_id [ Primary key ]

Po\_id [ Foreign key; references id of the purchase\_order table]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- carton\_id

Justification:carton\_id is the primary key and is used to refer the supplier\_dispatch\_carton table as it identifies each row uniquely.

**17. Table:**supplier\_dispatch\_carton\_item

**Columns:**

carton\_id [ Primary key,

Foreign key; references id of the supplier\_d’ispatch\_carton table]

Item\_id [ Foreign key; references id of the global\_item table ]

Quantity

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- carton\_id,item\_id

Justification:{carton\_id,item\_id} is the primary key and is used to refer the supplier\_dispatch\_carton\_item table as it identifies each row uniquely.

**18. Table:**supplier\_representative

**Columns:**

uid [ Primary key,Foreign key references id of the user table]

Supplier\_id [ Foreign key; references id of the supplier table]

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- uid

Justification:uid is the primary key and is used to refer the supplier\_representative

table as it identifies each row uniquely.

**Index 2:** Non clustered- Supplier\_id

Justification: indexing on Supplier\_id will help to display all the users/supplier\_representative of a particular supplier.

**19. Table:** Report

**Columns:**

carton\_id [ Foreign key; references id of the supplier\_dispatch\_carton table]

Item\_id [ Foreign key; references id of the global\_item table ]

report\_type

quantity

**Highest Normalization Level** : <4NF>

**Indexes:**

**Index 1:** Clustered- carton\_id,item\_id

Justification:{carton\_id,item\_id} is the primary key and is used to refer the report table as it identifies each row uniquely.

**Index 2:** Non clustered- report\_type

Justification: indexing on report\_type will help to display all the cartons of a particular type like - surplus or deficit.

**Part 3 : Stored programs**

1. **Trigger**: BEFORE INSERT trigger on store\_order table

Goal: Whenever a new store\_order is created by a store representative,the delivery date is automatically set to 7days from the current day.

DELIMITER //

CREATE TRIGGER before\_storeOrder\_insert

BEFORE INSERT ON store\_order

For each row

BEGIN

set

new.delivery\_date=DATE\_ADD(CURDATE(),INTERVAL 7 DAY);

END //

DELIMITER;

2. **Stored procedure**: report\_insert\_deficit

Parameters: IN carton\_id INT, IN item\_id INT,IN quantity int

Goal: This stored procedure is used to add entries into report table when there is deficit in the received order from supplier

DELIMITER //

CREATE PROCEDURE report\_insert\_deficit( IN carton\_id INT, IN item\_id INT,IN quantity int)

BEGIN

INSERT INTO retail1.report(carton\_id,item\_id,report\_type,quantity)

VALUES(carton\_id,item\_id,"DEFICIT",quantity);

END //

3. **Stored procedure**: report\_insert\_surplus

Parameters: IN carton\_id INT, IN item\_id INT,IN quantity int

Goal: This stored procedure is used to add entries into report table when there is surplus in the received order from supplier

DELIMITER //

CREATE PROCEDURE report\_insert\_surplus( IN carton\_id INT, IN item\_id INT,IN quantity int)

BEGIN

INSERT INTO retail1.report(carton\_id,item\_id,report\_type,quantity)

VALUES(carton\_id,item\_id,"SURPLUS",quantity);

END //

4. **Stored procedure**: GetUsername

Parameters: in id int(11)

Goal: This stored procedure is used to get the username for a given userid.

DELIMITER //

CREATE PROCEDURE GetUsername(in id int(11))

BEGIN

SELECT username FROM user where uid=id;

END //

DELIMITER ;

5. **Stored procedure**: GetDesignation

Parameters: in id int(11)

Goal: This stored procedure is used to get the designation for a given userid.

DELIMITER //

CREATE PROCEDURE GetDesignation(in id int(11))

BEGIN

SELECT designation FROM user where uid=id;

END //

DELIMITER ;

6. **Stored procedure**: GetItemName

Parameters: in id int(11)

Goal: This stored procedure is used to get the Item Name for a given userid.

DELIMITER //

CREATE PROCEDURE GetItemName(in id int(11))

BEGIN

SELECT name FROM global\_item where item\_id=id;

END //

DELIMITER ;

7. **Stored procedure**: GetItemDescription

Parameters: in id int(11)

Goal: This stored procedure is used to get the Item Description for a given userid.

DELIMITER //

CREATE PROCEDURE GetItemDescription(in id int(11))

BEGIN

SELECT description FROM global\_item where item\_id=id;

END //

DELIMITER ;

**Part 4 : Implementation**

SQL Dump for database : retail1 has been included with the submission.

**Part 5 : User interface**

The user interface has been created using JSP and servlets. Depending on different users, the pages are directed to different Servlet which takes care of the HTTP request and respond.