|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CMPT370 – GROUP 02 – FALL 2013 | | | | |
| Milestone 5: Warehouse Management System | | | | |
| Updated Design and Near-Complete Implementation | | | | |
|  | | | | |
| Xingze Guo xig480 | Rongli Han roh919 | Xianming Luo xil430 | Spencer Ondrusek spo798 | Matt Triff mdt879 |
|  | | | | |

1. **System Operations**

1. routeStockTasks(Object[][] itemsList)

Controller - Stock Handler

A list of items that must be gathered or put away are passed in, they are then ordered by the operation to provide the optimal path. The ordered list is passed back to the stock handler

controller.

2. locateProduct(int itemId)

Controller - System

An item ID is passed to this operation. The operation then locates the products current location(s) within the warehouse and returns the bin coordinates (x, y) back to the system.

3.Select shipping requirement:

User select a shipping requirement in table. Then presses ship. The new shipping task generated in system. and the data of item number, item name, item quantity, destination, data has been assign to the new shipping task

4.Assign shipping task to shipping company and stock handler:

User select a stock handler and a shipping company in the table. Then presses comfirm, system will assign the selected stock handler and shipping company to the new shipping task

5.Add new employee info

Manager click the add button in the bottom of Manager UI. the new window is visible and type in the new employees information in the related textfields. Then click save button to save and close table.

If the manager change their mind, he or she can click cancel.

6.edit employee info

Manager selects an employee in the current Employee List. the click edit button and open a new window. Similarly, managers can change and save it, or cancel it.

7. delete employee info

Manager selects employees info which isn’t needed anymore. then click delete button to delete it.

8. drawOnClickBin (int x, int y)

Controller - all users

add the new bin in the array and call repaint() function to draw the bin, which will display on the map system.

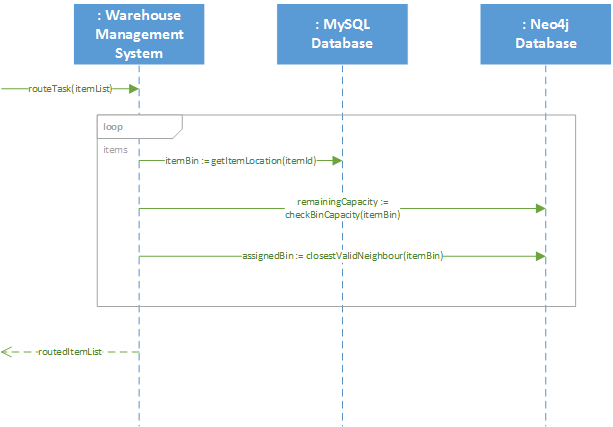
9. drawCoordinate (Graphics g)

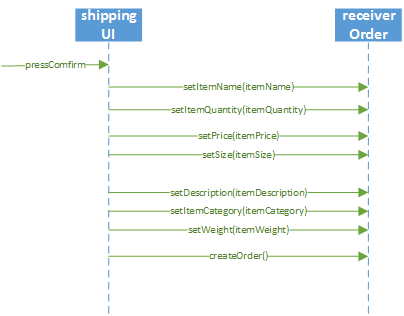
Controller - all users

draw the coordinate which points out which bin the user is working on right now.

1. **Interaction Diagrams/Sequence Diagrams**

*Routing Stocking Tasks – routeTask(itemList)*



* Confirm Products Received – confirmReceive(item)*

1. **Class Diagram**

*Login User Manual*

Currently, the system only has one user, to log in enter the credentials as follows:

User ID #: 1

Password: password

Permission: Manager

Any other credentials will deny your access to the system.

*Map User Manual*

1. Add bins: double click an empty cell.

2. Delete bins: right click a non-empty cell and click “delete” option on the shown popup.

3. Move bins: drag the bins around; if the coordinate gets red, it’s not allow to move the bins.

4. Locate bins: choose Menu bar (Search -> Locate) and enter the item id, then press locate button.

5. Check Details: double click a non-empty cell.

NOTE: Do not move the bins out of the map

*Database Package User Manual*

The database package utilizes two different databases, a MySQL Database, and a Neo4j Graph Database.

The MySQL database is located on the uSask servers, and is connected to remotely by the MysqlDB class.

The Neo4j database is stored locally within the application, in the graphDatabase/warehouse-db folder. No set up is required by the user to install or use Neo4j.

Calling each database to run a query is done through the static method runQuery, for example:

MysqlDB.runQuery(query);

Neo4jDB.runQuery(query);

Within the database package is a number of interfaces and classes that implement the database functionality for each major task. This are the only classes that will use the runQuery methods from above.

Only the classes in the sub-packages of the database package will ever access the MysqlDB.java and Neo4j.java classes. Each major function of the system has it's own database class that contains methods that will create queries, request the queries to be run, process and format the results, and return them to the calling method for use.

The 5 classes that can be used are:

GuiDB

ManagerDB

ReceiverDB

ShipperDB

StockHandlerDB

Within each class are all of the functions required for use by each part of the program. To use each class, an instance must be created, that instance can then be used to run the methods for each class.

Overall, this is done by:

<name>DB database=new <name>DB();

<returnType> return=database.<method>(<parameter1>, <parameter2>...<parameterN>);

Here is an example usage:

ManagerDB database=new ManagerDB();

String return=database.getProductName(123); //Where 123 is the product's ID

1. **Implementation**
2. **User Manual**
3. **Meeting Minutes**
4. **Git Log**
5. **Project Plan**