

**30 May 2025**

**Supermarket Inventory Management System for deli section**

**SIMSd**

PROJECT REPORT

**Student ID: 1602741**

**Chengfeng Li**

Table of Content

[Project Scope and Proposal 1](#_Toc31569)

[Stakeholders 2](#_Toc11977)

[Project Vision 2](#_Toc26251)

[Functionalities 2](#_Toc6835)

[Goal and Objectives 3](#_Toc13860)

[UML and Design 4](#_Toc7557)

[Project Design 5](#_Toc22007)

[UML of Data 6](#_Toc19917)

[UML of GUI 7](#_Toc31735)

[Functionality and Attribute Description 8](#_Toc1257)

[Purpose of attribute 11](#_Toc29297)

[UI Sample screens 12](#_Toc2224)

[User Guide 18](#_Toc410)

[Supplier Management 19](#_Toc7000)

[Items-From-Supplier (IFS) Management 25](#_Toc23135)

[Item-Produced-In-Store (IPS) Management 30](#_Toc27514)

[Report 36](#_Toc13511)

[Testing And Test Results 37](#_Toc28016)

[1. Create a supplier 37](#_Toc9844)

[2. Search suppliers by Name 37](#_Toc12380)

[3. Edit a supplier 38](#_Toc6651)

[4. Delete a supplier 38](#_Toc5264)

[5. View Report(suppliers) 39](#_Toc22058)

[Future Work 40](#_Toc13887)

# Project Scope and Proposal

**\*** This part of document is for the project ‘**Supermarket Inventory Management System for deli section (SIMSd)’** which includes the project idea, its main functionality, the goal and objectives of the project.

## Stakeholders

Product Owner: Supermarket Manager John Doe.

Project Manager: Peter Jackson

End User: Management Team, Sales Team, In-store items producer

## Project Vision

**For** management team and sales team

**Who** ‘d like to provide reliable products and improve customer’s satisfactions

**THE** Supermarket Inventory Management System for deli section (SIMSd.)

**IS A** powerful and easy-to-use tools running on any computer,

**THAT** tracking key products detail, like expiration dates, stocks and pricing and managing Suppliers.

**UNLIKE** the free-range records of piles of paper and receipt, SIMSd will speed up the statistic and decision making to match the fast-pacing market needs, also ensure compliance with food safety standards.

**For** in-store item producers

**Who** are providing fresh products.

**THE** SIMSd

**IS A** 24/7 instant stocks monitor

**THAT** helps to find out useful information, such as stocks and margin.

**UNLIKE** the traditional quantity and quality control system - guessing, SIMSd will reduce waste and improve efficiency.

With the development of ABC supermarket, SIMSd can be applied to all the branch network in the near future, strengthening the connection with supplier and customer.

## Functionalities

**Core Features**

1. In-store Items Management and Tracking: Add/Edit/Delete/Search.
2. From-Supplier items Management and Tracking: Add/Edit/Delete/Search.
3. Supplier Management - Add/Edit/Delete/Search
4. Expiration Alert - List items close to exp date/Dump/Promotion.
5. Stock & Waste Reporting – Report stock and waste’s quantity and values.

The system will differentiate two types of deli items:

1. In-Store Produced Items

e.g. daily-baked bread, fried chicken, etc.

* Item Name & Description
* Sell By Date (last day for sale)
* Use-By Date (last safe consumption date)
* Date Produced (when the item was made in-store)
* Price per Gram (variable pricing based on weight)
* Quantity in Grams (current stock weight)

**B.** Supplier-Provided Items

E.g. Yum Noodle, Farmer’s Choice Eggs, etc.

* Item Name & Description
* Sell By Date (last day for sale)
* Best Before Date (quality expiration)
* Price per Pack (fixed pricing per unit)
* Quantity in Packs (stock count)
* Supplier Information (vendor details)

## Goal and Objectives

**Goal**

Develop a reliable and efficient system for the deli section to track items fast, monitor stock and reduce waste.

**Objectives**

Reliable – System should run stably under high freq operations.

Accuracy – Ensure correct tracking with real-time updates.

User Friendly – Create a readable and friendly interface for end users.

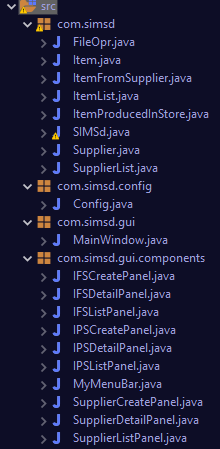
# UML and Design

**\*** The following part of document includes:

1. the project design,
2. UML of used classes,
3. functionality of each method,
4. the purpose of each attribute,
5. Sample screen of UI.

## Project Design

SIMSd will be developed in JAVA language with Swing for GUI, and Data will be stored in local JSON file. It will be a structured program that follow enforced programming language conventions and programming standards. Source file and package structure are as figure below:



com.simsd : for major class files of app

com.simsd.config: System config files

com.simsd.gui: App Window GUI files

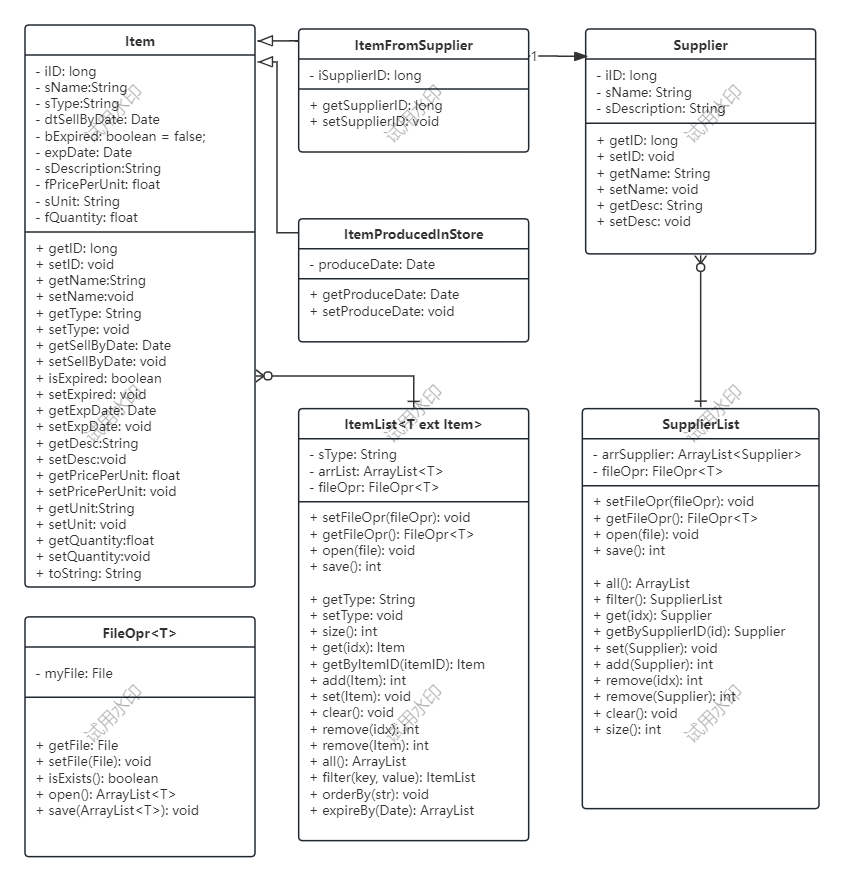
com.simsd.gui.components: GUI components

For Learning Outcome 4, **ArrayList** are wildly used in app as main Data Structure.

When starts, class **FileOpr** helps App to load the data from 3 local JSON files into class **Items** and class **Supplier**. There are 2 type of item, **ItemFromSupplier** and **ItemProducedInStore** which are both inherited from class **Item**. All loaded items are organized and stored in List, **ItemList** and **SupplierList** for app further use.

For functional, There’d be CRUD operations for both kind of Items and Supplier. Every time data modified, it will be writed into the JSON via FileOpr. As well , basic search functions are provided.

## UML of Data



## UML of GUI



## Functionality and Attribute Description

Below are the description of **some major** class: FileOpr, ItemList and SupplierList, MainWindow, IFSListPanel, IFSDetailPanel.

**FileOpr**

Attr:

1. File myFile: the local data JSON file.

Function:

1. getFile() and setFile(File): the getter and setter of variable myFile.
2. isExists(): check if myFile exists or not. Return true if the myFile exists.
3. Open(File): open and read the file as specified data type, return an ArrayList of them.
4. Save(ArrayList): save the input ArrayList into myFile.

**ItemList**

Attr:

1. sType: Indicate what type of item the ItemList instance will hold.
2. arrList: The ArrayList that holds the item.
3. fileOpr: the File Operater instance that read and save the items data from local JSON.

Function:

1. ItemList(File, type): The Constructor will initialise the FileOpr and read the JSON for all the item of specified type.
2. Open(): let the fileOpr to read data into proper type arrList.
3. Save(): let the fileOpr to write arrList into proper JSON
4. Add(Item): add item into arrList
5. Remove(inx): remove item from arrList by index
6. Remove(Item): remove item from arrList by item
7. Clear(): empty the arrList.
8. getByItemID(id): return item from arrList by id
9. Get(int): return item from arrList by index
10. All(): return all items from arrList
11. Size(): return amount of items in arrList
12. Update(int, item): update the item in arrList by index
13. Update(item): update the item in arrList by itself
14. Filter(key, value): search items by key and value in arrList.
15. expireBefore(Date): return items resulte set that’d expired before a certain date.

**SupplierList**

Attr:

1. arrSupplier: The arraylist that holds the suppliers.
2. fileOpr: the File Operator instance that read and save the suppliers data from local JSON.

Function:

1. SupplierList(File): The Constructor which will read suppliers data from File.
2. Open(): let the fileOpr to read data into arrSupplier.
3. Save(): let the fileOpr to write data into JSON.
4. Add(Supplier): add supplier into arrSupplier.
5. Remove(index): remove supplier from arrSupplier by index.
6. Remove(Supplier): remove the specified supplier from arrSupplier.
7. Clear(): clean all suppliers from arrSupplier.
8. Get(int): get supplier from arrSupplier by index.
9. getBySupplierID(long): get supplier from arrSupplier by supplier ID
10. All(): return all suppliers in the arrSupplier.
11. Size(): return arrSupplier’s size.
12. Filter(key, value): search supplier by key and value in arrSupplier.
13. Set(Supplier): update the specified supplier.

**MainWindow**

Attr:

1. - instance: MainWindow //the only instance of MainWindow
2. + supplierList: The **global** Supplier List for the app
3. + IFSList: The **global** ItemFromSupplier List
4. + IPSList: The **global** ItemProduceInStore List
5. + mainPanel: JPanel

// following 10 panel are the 10 functional page of app, keeping in a CardLayout and ready to be switched any time by clicking the menu.

1. + supplierListPage: SupplierListPanel
2. + supplierDetailPage:SupplierDetailPanel
3. + supplierCreatePage: SupplierCreatePanel
4. + ifsListPage: IFSListPanel
5. + ifsDetailPage: IFSDetailPanel
6. + ifsCreatePage: IFSCreatePanel
7. + ipsListPage: IPSListPanel
8. + ipsCreatePage: IPSCreatePanel
9. + reportPage: ReportPanel
10. + cardLayout: CardLayout.

Function:

1. - MainWindow(): Private Constructor, ensure that only one instance of MainWindow.
2. + getInstance() : Public getter of the instance.
3. - createPage(): initialize all the pages and store them in CardLayout.
4. - initDate(): Read data from JSON file.
5. + switchToPage(String toView): Switch the main view to the requested page.

**IFSListPanel**

Attr:

1. - mainWindow: MainWindow for reference.
2. - table: JTable component to display the data result.
3. - tableModel: Keeping the data to be displayed.
4. - schKey: Holding the search key of “Name” or “Description”.
5. - schValue: User entered search value.
6. - dataList: All the ItemFromSupplier from JSON, passed in from MainWindow.
7. + displayList: ItemFromSupplier to be displayed, either search result or dataList.

Function:

1. + IFSLIstPanel(MainWindow): Constructor, init the GUI according to the datalist
2. + doSearch(): Search in dataList and return a new List as display list.
3. + reLoadTable(): reload JSON into datalist and display.
4. + saveTable(): Call the ItemList.save(), to save the datalist into JSON.
5. + setDataList(ItemList): Setter of datalist.

**IFSDetailPanel**

Attr:

1. - mainWindow: MainWindow for reference
2. - ifsItem: Reference to the current ItemFromSupplier.
3. - txtName, txtDesc, txtType, txtSellByDate, txtExpDate, txtPricePerUnit, txtUnit, txtQuantity: JTextField Components for displaying and editing IFS’s information.
4. - supplierComboBox: JComboBox components for displaying and editing supplier’s list.

Function:

1. + IFSDetailPanel(MainWindow): Constructor, initialize the GUI.
2. + setItem(ItemFromSupplier): initialize the data of current IFS.
3. + delete(): Prompt the confirmation, then delete current IFS from datalist and save to JSON.
4. + jumpToList(): Switch to list page.
5. + save(): valid the input, then update the current IFS, and save to JSON.
6. + validInput(): If the input info is valid, return true.

## Purpose of attribute

We’ll descript some major attributes of class: Supplier, Item, ItemFromSupplier and ItemProducedInStore.

**Different kinds of relationship**:

* ItemFromSupplier and ItemProducedInStore are inheritated from Item.
* Supplier and ItemFromSupplier is one-to-one Foreign Key.

**Supplier**

1. Long iID: auto created supplier id, it’s actually the datetime in which the supplier created.
2. String sName: Supplier’s name.
3. String sDescription: Description of the supplier.

**Methods in this class are only getters and setters.**

**Item**

1. Long iID: auto created item id, it’s the datetime that item created.
2. String sName: Item’s name.
3. String sType: Item’s type.
4. Date dtSellByDate: date by which the item should be sold
5. Bool bExpired: Is the item expired.
6. String sDescription: item’s description.
7. Float fPricePerUnit: item’s unit price.
8. String sUnit: item’s unit
9. Float fQuantity: item’ s stocked quantity.

**Methods in this class are only getters and setters.**

**ItemFromSupplier extends Item**

1. iSupplierID: foreign key, supplier’s id.

**Methods in this class are only getters and setters.**

**ItemProducedInStore extends Item**

1. Date produceDate: the date that item was produced.

**Methods in this class are only getters and setters.**

## UI Sample screens

1. Index Page and Menu.

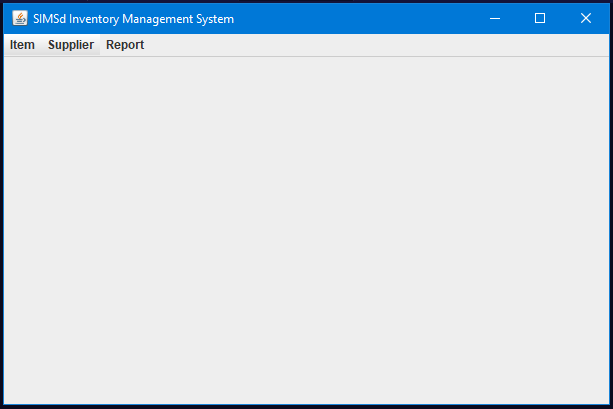


Fig 1.0 Index Screen

|  |  |
| --- | --- |
| Fig 1.1 Item From Supplier Menu | Fig 1.2 Item Produced In Store Menu |
| Fig 1.3 Supplier Menu |  |

1. Item From Supplier - List, Create, Edit, Delete

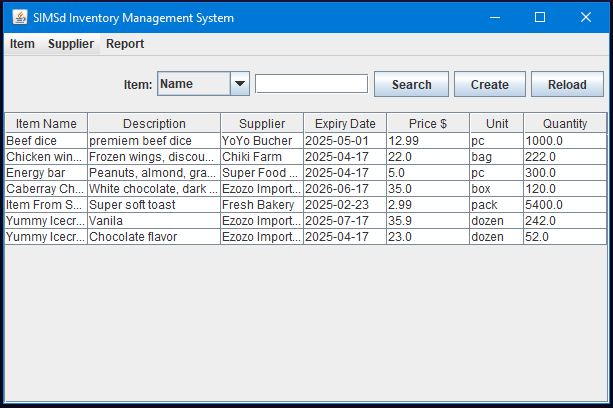


Fig 2.0 Item From Supplier List

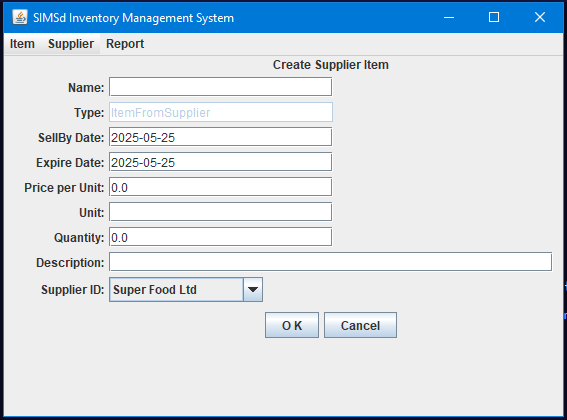


Fig 2.1 Item From Supplier Create

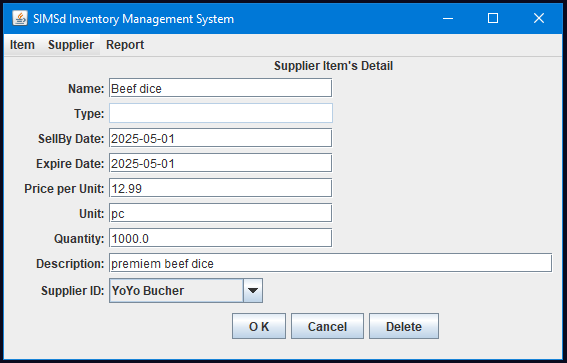


Fig 2.2 Item From Supplier Edit

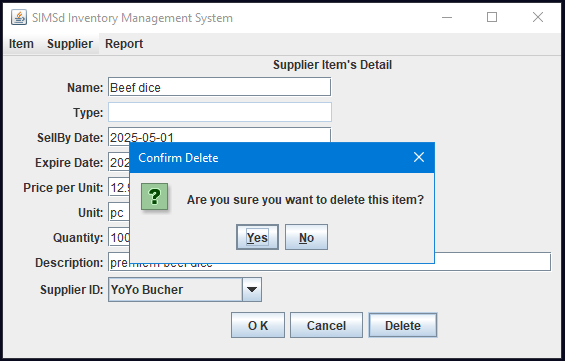


Fig 2.3 Item From Supplier- Delete Confirm

1. Item Produced In Store - List, Create, Edit, Delete

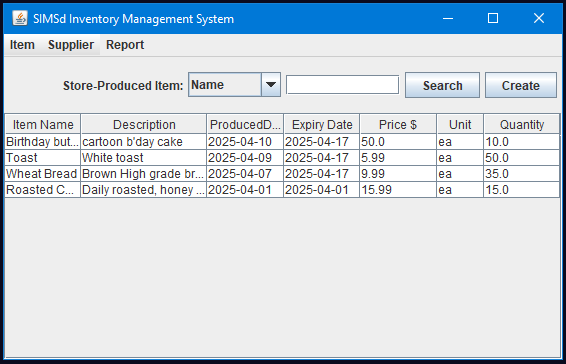


Fig 3.0 Item Produced in Store - List

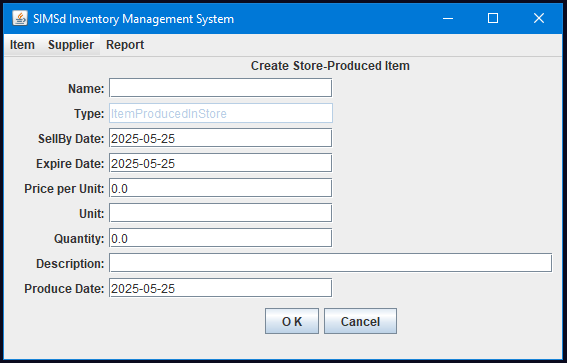


Fig 3.1 Item Produced in Store - Create

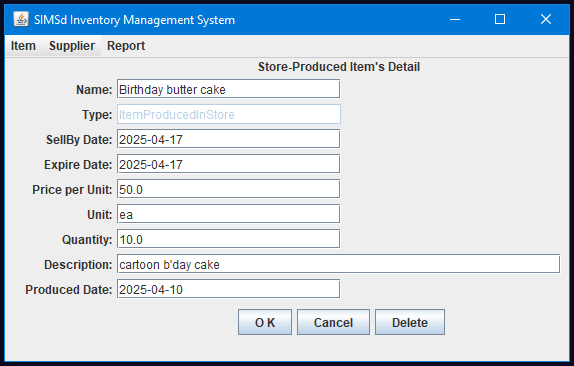


Fig 3.2 Item Produced in Store - Edit

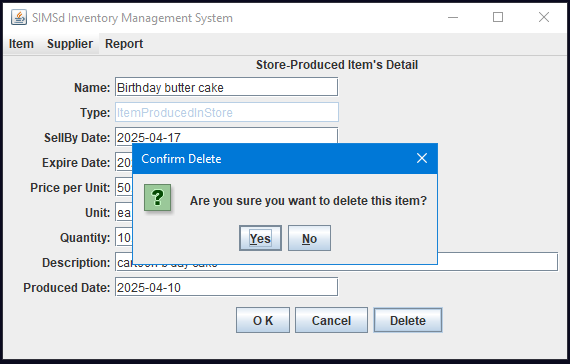


Fig 3.3 Item Produced in Store - Delete confirm

1. Report

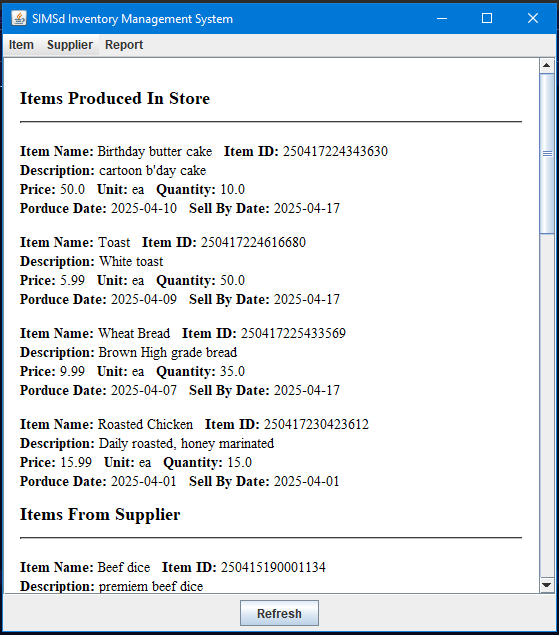


Fig 4.0 Report

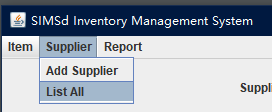
# User Guide

\* User guide helps new user to get familiar with and use the app. Screen shot are available along with operation steps.

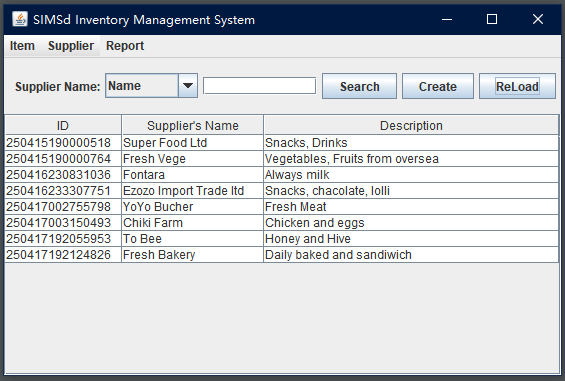
## Supplier Management

1. List all suppliers

Operate: [Supplier]-[List All]

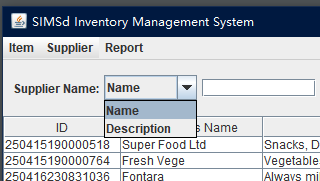


Result:

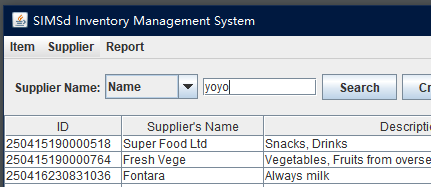


1. Search suppliers by Name or Description

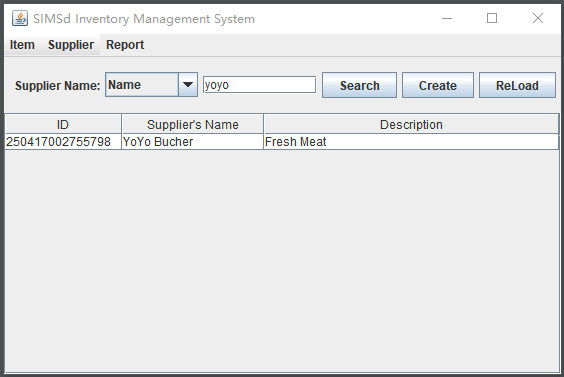
Op1: Select “Name”, or “Description”.



Op2: Type in “yoyo” as key word, then click “Search” button,

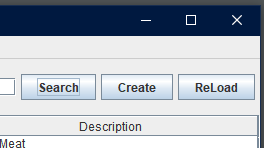


Result: Search supplier whose Name is like “yoyo”, ignore case.

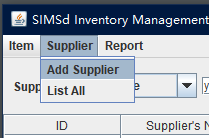


1. Create a supplier

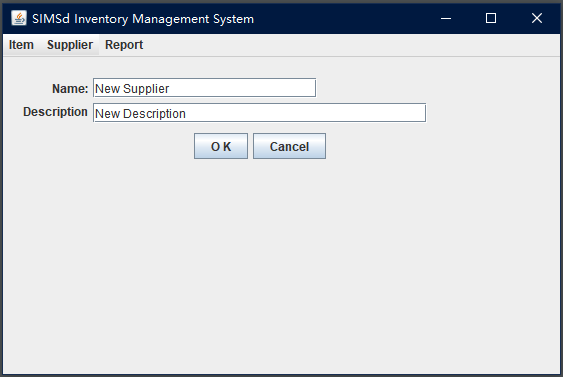
Op1A: Menu[Supplier]-[List All]-[Create]



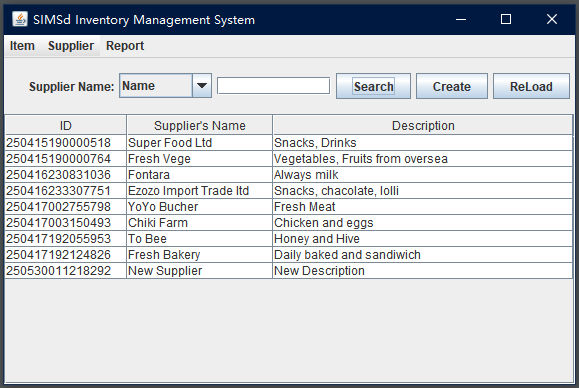
Op1B: Menu[Supplier]-[Add Supplier]



Op2: Type in “New Supplier” for Name, and “New Description” for Description, then click “OK”,

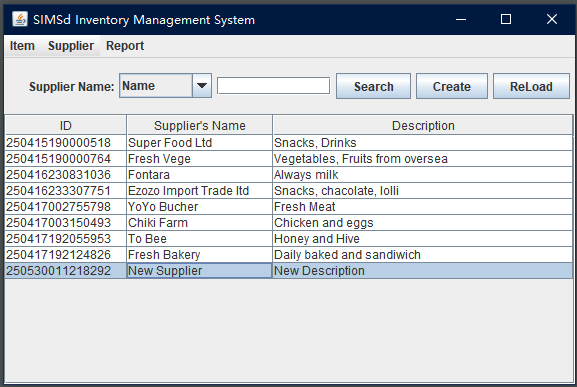


Result: “New Supplier” will be showed in the list.

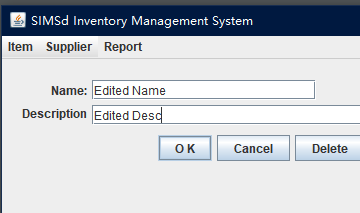


1. Edit a supplier

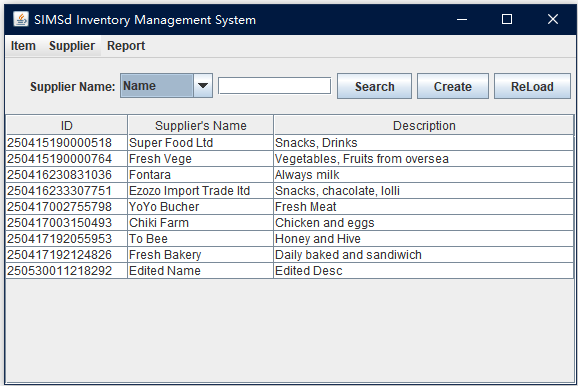
Op1: Double click “New Supplier” in the list.



Op2: Type in “Edited Name” for Name, and “Edited Desc” for Description, then click “OK”.



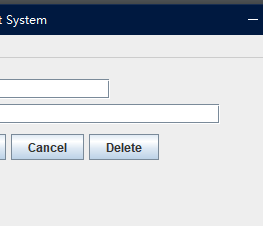
Result: Modified supplier is showed in the list.



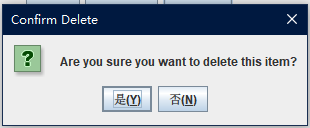
1. Delete a supplier

Op1: Double click the supplier needed to be deleted, eg. “Edited Name”.

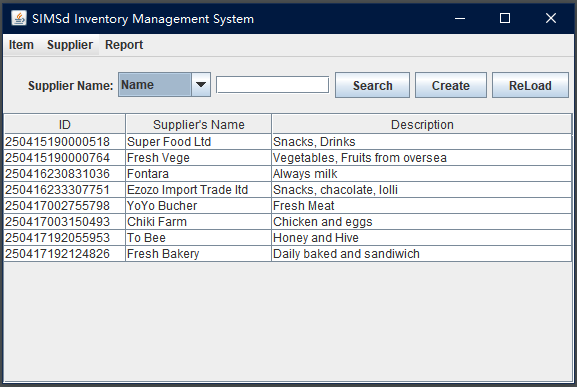
Op2: Click “Delete” button.



Op3: Click “Yes”.



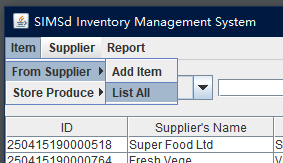
Result: supplier is removed from the list.



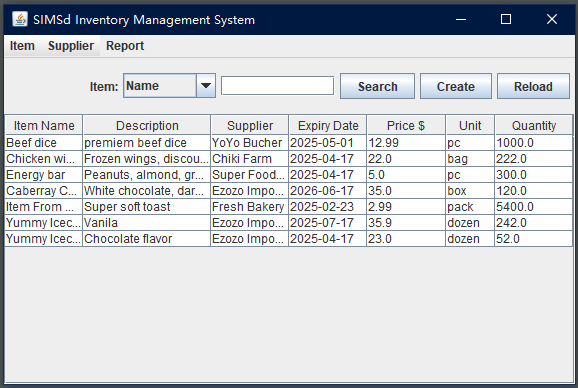
## Items-From-Supplier (IFS) Management

1. List All IFS

Op1: Menu[Item] - [From Supplier] - [List All]



Result: Show List of all IFS

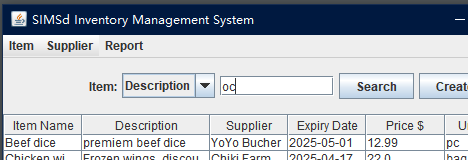


1. Search IFS

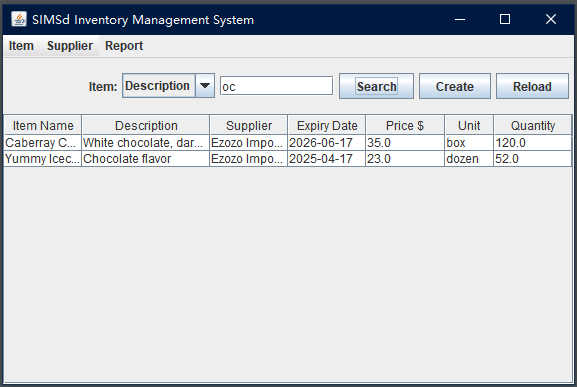
Op1: Select “Description” or “Name” as key,

Op2: Type in “oc” as value,

Op3: Click “Search” button.



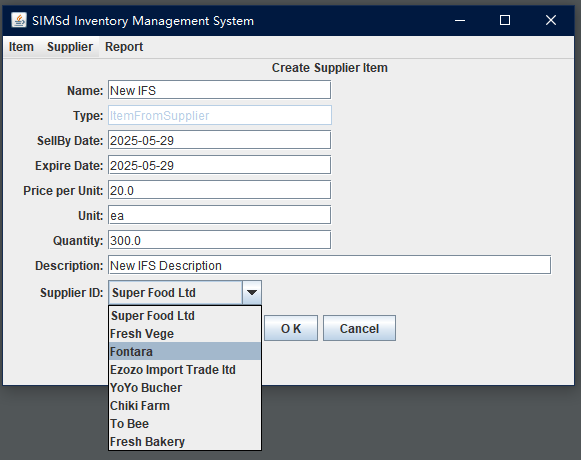
Result: Show all IFS whose key includes value of “oc”.



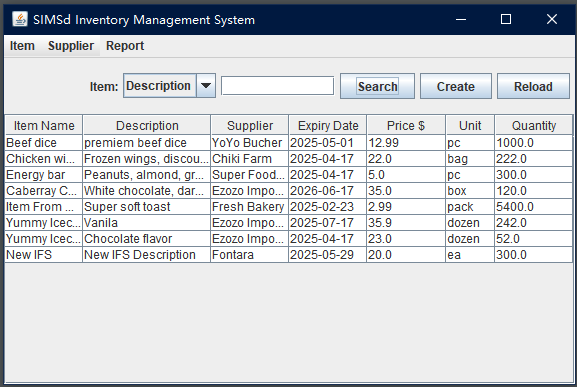
1. Create an IFS

Op1: Menu[Item] - [From Supplier] - [Add Item]

Op2: Type in info of new IFS, select “Fontara” as Supplier, then “OK”,



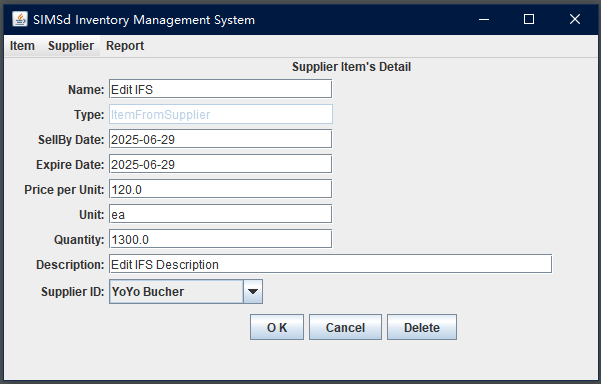
Result: “New IFS” is shown in the list with info entered.



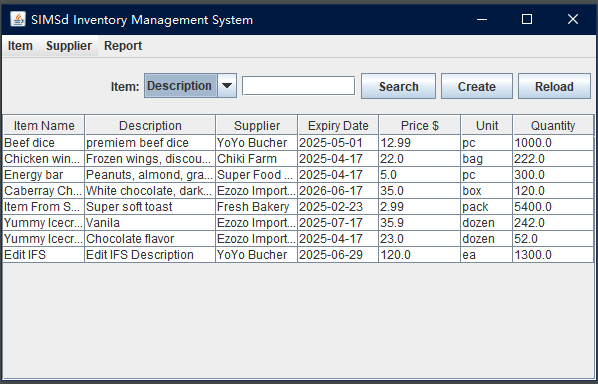
1. Edit an IFS

Op1: Double click the listed Item needed to be edited, like “New IFS”.

Op2: Type in new info, then “OK”.



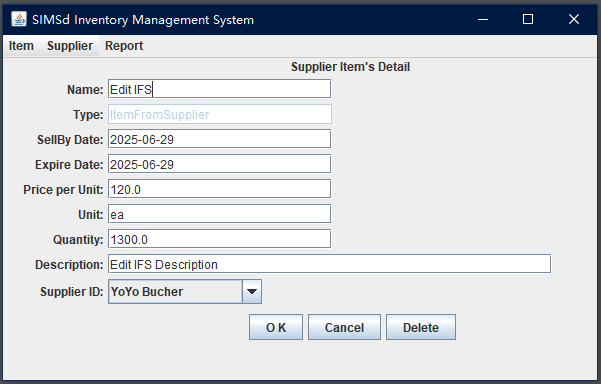
Result: Edited IFS is shown in the list.



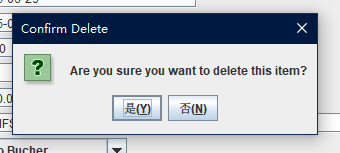
1. Delete an IFS

Op1: Double click the IFS needed to be deleted, like “Edit IFS”.

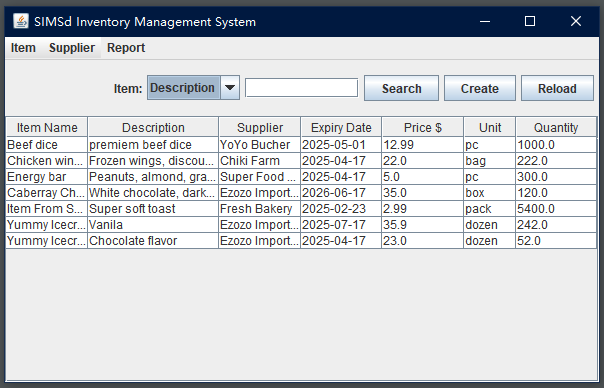
Op2: Click the “Delete” button,



Op3: Click “Yes” to confirm.



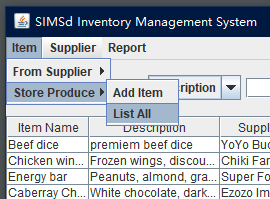
Result: IFS “Edit IFS” is removed from the list.



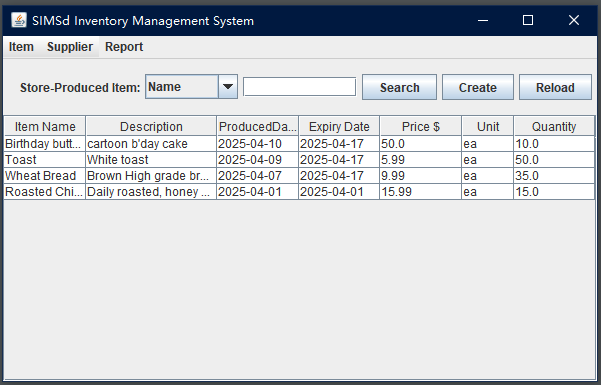
## Item-Produced-In-Store (IPS) Management

1. List All IPS

Op1: Menu[Item] - [Store Produce] - [List All]



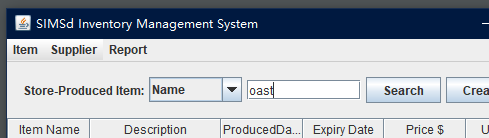
Result: List all of IPS



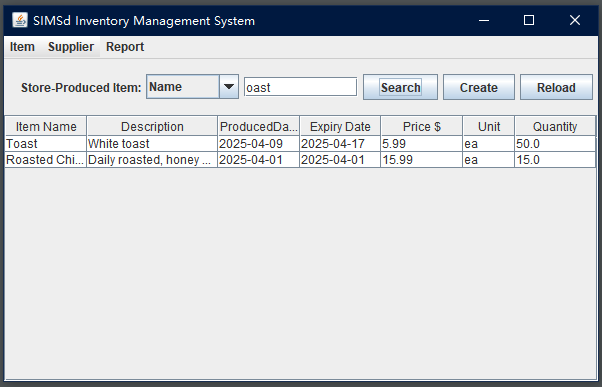
1. Search IPS

Op1: Select “Name” or “Description” as key,

Op2: Type in “oast” as value, then “Search”.

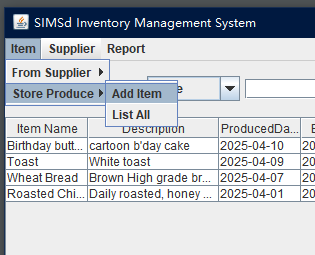


Result: return a list of whose key includes value of “oast”.

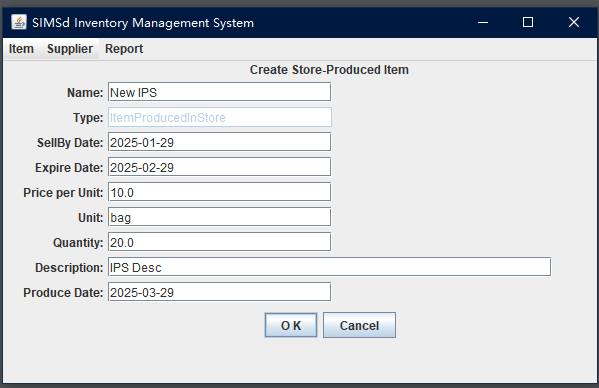


1. Create an IPS

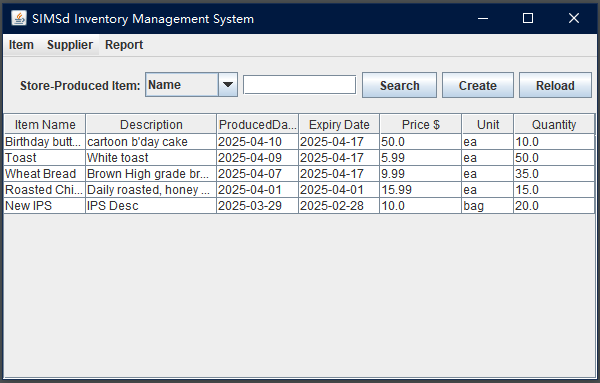
Op1: Menu[Item] - [Store Produce] - [Add Item]



Op2: Type in info needed, then “OK”.



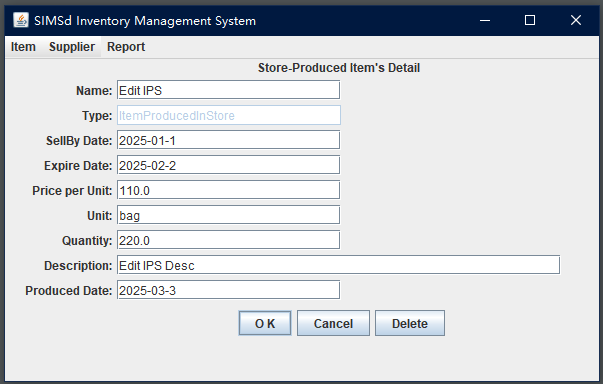
Result: “New IPS” is shown in the list.



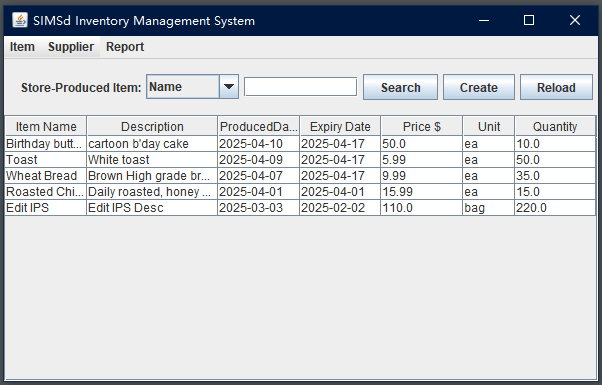
1. Edit an IPS

Op1: Double click the IPS want to be edited, like “New IPS”.

Op2: Type in new info, then “OK”,



Result: IPS updated and shown in the list.

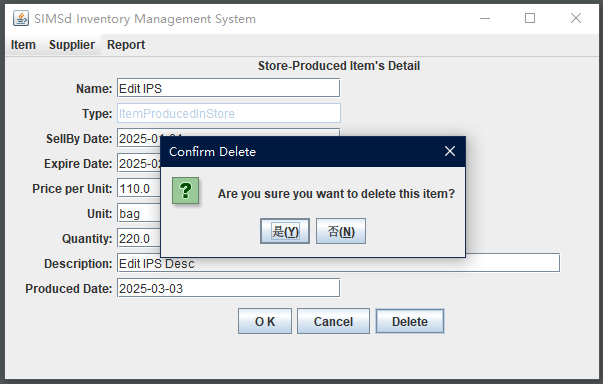


1. Delete an IPS

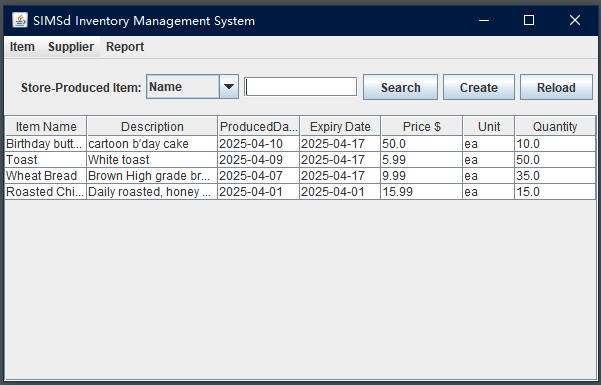
Op1: Double click the IPS want to be deleted, like “Edit IPS”,

Op2: Click “Delete”,

Op3: Click “Yes”



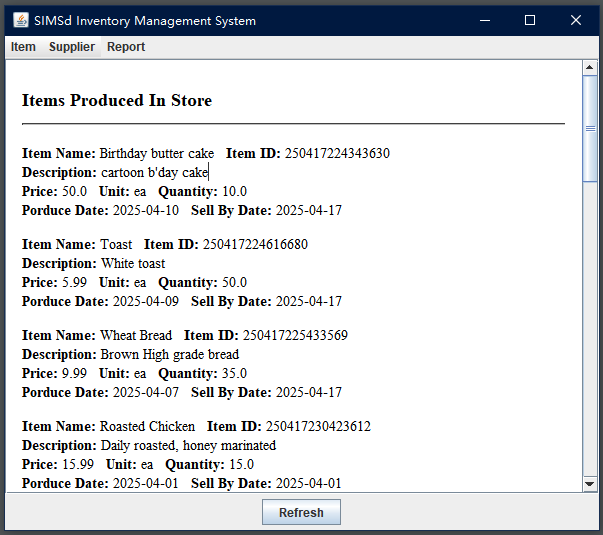
Result: IPS is successfully removed from the list.



## Report

Op1: Menu[Report]

Result: A report list of IPS, IFS and Supplier is shown on screen.



# Testing And Test Results

\* The following are 5 selected test cases for 5 testing scenarios.

## Create a supplier

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Steps** | **Expected Result** | **Status** |
| User can create a new supplier | 1. Menu[Supplier] - [Add Supplier] | Show create supplier page | **Passed** |
| 1. Click “OK” with blank Name | Err msg popped. | **Passed** |
| 1. Type in information |  |  |
| 1. Click “OK” | Back to supplier list page, new supplier is shown in the list. | **Passed** |

## Search suppliers by Name

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Steps** | **Expected Result** | **Status** |
| User can Search suppliers by supplier’s Name or Description, results are shown in list | 1. Menu[Supplier] - [List All] | Show all supplier in list | **Passed** |
| 1. Select “Name”, type in “@”, then click Search. | Empty list is shown | **Passed** |
| 1. Clear search value, then click Search | Show all supplier in list | **Passed** |
| 1. Select “Name”, type in “o”, then “Search”. | Return suppliers with at lease one “o” in the Name | **Passed** |
| 1. Select “Description”, type in “ea”, then “Search”. | Return suppliers with at lease one “ea” in the Description.s | **Passed** |

## Edit a supplier

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Steps** | **Expected Result** | **Status** |
| User can edit a supplier’s information. | 1. Menu[Supplier] - [List All] | Show supplier list | **Passed** |
| 1. Double click a supplier | Open edit page with correct supplier’s info | **Passed** |
| 1. Click “Cancel” | Return the previous list | **Passed** |
| 1. Double click a supplier again |  |  |
| 1. Type new info for this supplier, then click “OK” | Return to previous list, the supplier’s new info is shown. | **Passed** |

## Delete a supplier

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Steps** | **Expected Result** | **Status** |
| User can delete a supplier | 1. Menu[Supplier] - [List All] | Show all supplier | **Passed** |
| 1. Double click a supplier | Show supplier’s detail page | **Passed** |
| 1. Click “Delete” | Confirm msg is popped out | **Passed** |
| 1. Click “Cancel” | Confirm msg is closed | **Passed** |
| 1. Click “Delete” again |  |  |
| 1. Click “Yes”. | Return to supplier list, the supplier is removed. | **Passed** |

## View Report(suppliers)

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Steps** | **Expected Result** | **Status** |
| User can view a report that include all supplier’s info. | 1. Menu[Report] | Report page is shown | **Passed** |
| 1. Scroll down to “Suppliers” | View suppliers info | **Passed** |

# Future Work

To continuously bring value to customer is the essential goal of our app, improvement will be achieved in future upgrades with new features, including but not limited to, as below:

* + - 1. Authentication,
      2. Registered user management,
      3. Expiration Alert
      4. Stock & Waste Reporting
      5. Refactoring and restructure
      6. Shift to database for data store,
      7. Improve with stock-take feature
      8. Facility to on-line or Cloud version
      9. Multi-store management

**Thank you**

Chengfeng Li

1602741