

CSCI222 – System development

Assignment 2 – warehouse management tool

Version 0.8



|  |  |  |
| --- | --- | --- |
| **Name** | **Student ID** | **Email** |
| Chan Xiaoru Kalista | 5986345 | xkchan002@mymail.sim.edu.sg |
| Tam He Sheng (Allen) | 5986485 | hstam001@mymail.sim.edu.sg |
| Teo Heong Hwee (Gavin) | 6098058 | hhteo005@mymail.sim.edu.sg |
| Jonas Chok Wen Jie | 6098459 | jwjchok001@mymail.sim.edu.sg |
| Kyaw Myo Aung, Johns | 6097868 | myoak001@mymail.sim.edu.sg |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Effective Date | Summary of Change | Prepared by |
| 0.1 | 07/02/2018 | Initial Draft | Johns |
| 0.2 | 10/02/2018 | Overview of System | Kalista |
| 0.3 | 10/02/2018 | Added:   * Project Timeline * Use Case Diagram * Detailed Use Case * Sequence Diagram * Meeting Minutes | Johns |
| 0.4 | 11/02/2018 | Added:   * Detailed Use Case Diagram * Sequence Diagram * Use Case Description | Heong Hwee |
| 0.5 | 18/02/2018 | Added:   * Data Dictionary * Class Diagram | Allen |
| 0.6 | 19/02/2018 | Added:   * Functional & Non-Functional Specs * User Interface Design * Functionalities of WMS * Test Cases & Report * Risk and Counter Measures | Kalista |
| 0.7 | 19/02/2018 | Updated   * Objective * Project Timeline * Responsibility Matrix * Login Sequence diagram | Johns |
| 0.8 | 20/02/2018 | Updated   * Use Case Diagram * Detailed Use Case * Sequence Diagram | Heong Hwee & Jonas |

Table of Contents

[1 Introduction 5](#_Toc506832881)

[1.1 Objective 5](#_Toc506832882)

[1.2 Definitions, Acronyms and Abbreviations 5](#_Toc506832883)

[2 Roles and Responsibilities 6](#_Toc506832884)

[2.1 Responsibility Matrix 6](#_Toc506832885)

[2.2 Role Allocation 7](#_Toc506832886)

[3 Implementation Plan 0](#_Toc506832887)

[3.1 Project Schedule 0](#_Toc506832888)

[4 Overview of System 1](#_Toc506832889)

[4.1 Functional Specifications 1](#_Toc506832890)

[4.2 Non-Functional Specifications 5](#_Toc506832891)

[4.3 User Interface Design 6](#_Toc506832892)

[4.3.1 Screen Designs (Prototype) 6](#_Toc506832893)

[4.4 Functionalities of WMS 8](#_Toc506832894)

[4.4.1 Login 8](#_Toc506832895)

[4.4.2 Control Panel 9](#_Toc506832896)

[4.4.3 Category Management 10](#_Toc506832897)

[4.4.4 Listing Management 16](#_Toc506832898)

[4.4.5 Movement Management 21](#_Toc506832899)

[4.4.6 Summary Report 23](#_Toc506832900)

[4.4.7 Logout 24](#_Toc506832901)

[4.5 Use Case Diagram 25](#_Toc506832902)

[4.5.1 Iteration 1 25](#_Toc506832903)

[4.5.2 Iteration 2 26](#_Toc506832904)

[4.6 Use Case Detailed Description 28](#_Toc506832905)

[4.6.1 Detailed Use Case – Login 28](#_Toc506832906)

[4.6.2 Detailed Use Case – Manage Stock Category 29](#_Toc506832907)

[4.6.3 Detailed Use Case – Manage Stock Movement 31](#_Toc506832908)

[4.6.4 Detailed Use Case – Manage Stock Listing 32](#_Toc506832909)

[4.6.5 Detailed Use Case – View Summary Report 33](#_Toc506832910)

[4.7 Sequence Diagram 34](#_Toc506832911)

[4.7.1 Sequence Diagram – Login 34](#_Toc506832912)

[4.7.2 Sequence Diagram – Manage Stock Category 35](#_Toc506832913)

[4.7.3 Sequence Diagram – Manage Stock Movement 36](#_Toc506832914)

[4.7.4 Sequence Diagram – Manage Stock Listing 37](#_Toc506832915)

[4.7.5 Sequence Diagram – View Summary Report 38](#_Toc506832916)

[4.8 Class Diagram & Description 39](#_Toc506832917)

[4.8.1 Class Diagram Iteration 1 39](#_Toc506832918)

[4.8.2 Class Diagram Final Version 40](#_Toc506832919)

[4.8.3 Class Description 42](#_Toc506832920)

[5 Data Dictionary 44](#_Toc506832921)

[6 Risk and Counter Measures 46](#_Toc506832922)

[7 Meeting Minutes 47](#_Toc506832923)

# Introduction

## Objective

The warehouse Management system (WMS) allow users to manage, monitor the incoming, outgoing & balance stocks items in the inventory lists.

We designed the system in a way that has to be simple & user-friendly.

In this project we especially focus on the brainstorming and designed base on the client’s business needs and functional requirements.

We spent so much time to analyses the system on various researches.

The WMS tools will serve the following function in an efficient & effective way:

* Able to record the incoming, outgoing stocks list.
* Able to edit and track the existing stocks base on category and sub-category.
* Able to generate daily, weekly & monthly inventory summary report of stocks details
* Provide secure login encryption authentication process for the users.
* Able to search in stocks items base on date, time & prices
* Use the password encryption method for secure login process
* And many more useful features for the users and their business.

## Definitions, Acronyms and Abbreviations

**Acronyms and Abbreviations to be referenced from this table:**

|  |  |
| --- | --- |
| **Abbreviation/Acronyms** | **Definitions** |
| SRS | Software Requirement Specification |
| WMS | Warehouse Management System |
| QC | Quality Control |

# Roles and Responsibilities

## Responsibility Matrix

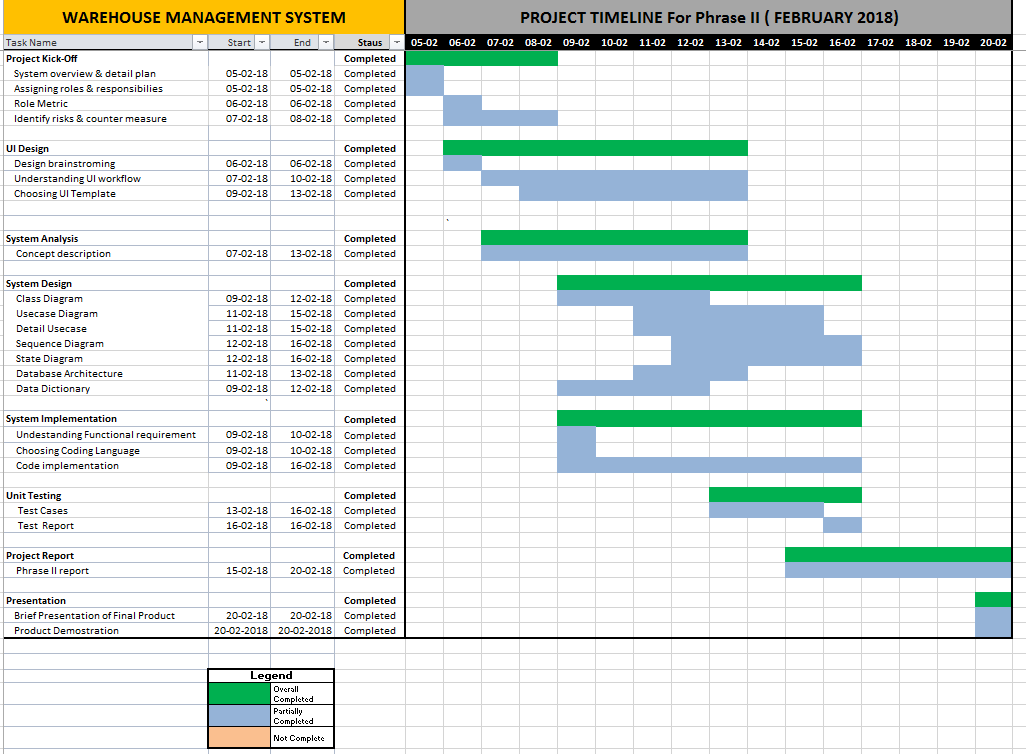
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Responsibilities | Name | | | | |
| Kalista | Allen | Gavin | Jonas | Johns |
| Project Schedule |  |  |  |  | **X** |
| Class Diagram and Description |  | **X** |  |  |  |
| Project Report | **X** | **X** | **X** | **X** | **X** |
| Coding & Implementation | **X** |  |  |  |  |
| Test Cases | **X** |  |  |  |  |
| Data Dictionary |  | **X** |  |  |  |
| Use Case Diagram/Description |  |  | **X** | **X** | **X** |
| Sequence Diagram/Description |  |  | **X** | **X** | **X** |
| Risk Assessment | **X** |  |  |  |  |
| UI Design | **X** |  |  |  |  |
| Overall Project Review | **X** | **X** | **X** | **X** | **X** |

## Role Allocation

| **Name | Title** | **Role** | **Responsibilities** |
| --- | --- | --- |
| Allen | | The Project Manager is responsible for developing, in conjunction with the Project Sponsor, the project charter. The Project Manager ensures that the project is delivered on time, within budget, and to the required quality standards. | * Manage and lead the project team. * Manage the coordination of the partners and the working groups. * Develop and maintain a detailed project plan. |
| Gavin | Technical writer | Technical writer prepares the supporting documents & instruction manuals | * Provide manuals &   System reference documents |
| Johns | System Analysis | System analysis evaluate system workflow & product specifications. | * Provide the overall workflow & concept |
| Kalista | Developer | The developer is the one who implement the system according to system design & analysis workflow. | * System implementation * Coding& testing of the products |
| Jonas | Lead System Designer  Jonas | Lead System Designer | System designer design the system based on the business needs & functional requirement | * Design interfaces & functionality to include in system |

# Implementation Plan

## Project Schedule



# Overview of System

## Functional Specifications

1. **Login**

This module allows the user to login and the first page that the user will see upon login is the Stock Control Panel. System will also validate the encrypted password and will lock the account if the user has made 3 unsuccessful attempts, this will result in the user not being able to login once the 3 attempts have been used up.

**Below depicts the processing logic for this function:**

1. User key in username and password.
2. System validate user is not locked and compare password.
3. If user is not locked then go to step 4 else show error message and end.
4. If password matches then save user session, redirect to Stock control panel, and end.
5. If password does not match then add 1 try to user attempt and show error message and end.
6. **Main Menu**

This module will display a main menu at the left side with the following menu option.

* 1. Control Panel
  2. Category Management
  3. Listing Management
  4. Movement Management
  5. Summary Report
  6. Logout

System redirects to module page when user clicks on the specific.

1. **Control Panel**

This module displays the stock items that quantity has fallen below the configured threshold. This is the first page that the user will see once successfully logged in.

1. **Category Management**

This module allows the user to manage the stock category of the system. The module will have a textbox to key in the category description and user will be able to select the parent category if any. If the parent category is not selected then the category will be set as a parent category.

* 1. Description
  2. Parent Category

Below depicts the processing logic for this function.

1. Load all category to parent category selector where ParentID = 0.
2. User key in category description and select parent category.
3. If parent category is not selected then set parent as 0.
4. If parent category is selected then set parent to selected value ParentID.
5. Save the entry when user click submit.
6. **Listing Management**

This module allows the user to manage the different stock item that the warehouse holds. User will have to create the stock first before any stock movement; stock in or stock out can be performed. User is able to Add/Edit the stock item, below states the fields for the stock item. It also allows the user to enter threshold for each stock item. For each stock item id, there will only be 1 threshold.

Add New Stock Listing

1. **Stock Description**
2. **Main Category**
3. **Sub-Category**
4. **Unit Price**
5. **Threshold**
6. **Count [Default 0]**
7. **Created On**
8. **Created By**
9. **Modified On**
10. **Modified By**

**Below depicts the processing logic for this function.**

* 1. **Add Stock Item**

1. User enter all required field.
2. User click on submit button.
3. System save record on submit.
   1. **Edit Stock Item**
4. User select stock item to edit.
5. User update all record field.
6. User click on submit button.
7. System save record on submit.
   1. **Search Stock item**
8. User key in search criteria.
9. User click on filter button.
10. System retrieve all match records on submit.
11. **Movement Management**

This module allows the user to enter stock movement into the system to keep track of current stock level real time.

* 1. **Stock Item**
  2. **Inbound/Outbound**
  3. **Count**

**Below depicts the processing logic for this function.**

* 1. **Inbound Stock**

1. User select stock item from the list.
2. User select inbound from the list.
3. User key in the stock movement count.
4. User click on submit button.
5. System save record on submit and update stock count for that stock item.
   1. **Outbound Stock**
      1. User select stock item from the list.
      2. User select outbound from the list.
      3. User key in the stock movement count.
      4. User click on submit button.
      5. System check user keyed in stock movement count is less than system stock count.
      6. System save record on submit and update stock count for that stock item.
6. **Stock Summary Report**

This module allows the user to generate reports based on the criteria selected. This allow user the flexibility to generate daily, weekly, monthly and yearly reports. User will be able to enter the Date from and Date to from the date picker.

The report will have the below columns:

* 1. **Transaction Date**
  2. **Inbound/Outbound**
  3. **Quantity**
  4. **Stock Item**
  5. **Unit Price**
  6. **Total Price**

## Non-Functional Specifications

1. **Maintainability**

The system have been coded in a way that is easy to maintain and will support scalability so that the system can grow and features can be added, functionalities to be further improved whenever necessary. The css styling is done in a generic way such that certain styles can be reused (e.g. shortTextBox has the same css styling).

1. **Usability**

The system is user-friendly and intuitive such that the design for the menu options are at the left hand side, which allows the user to navigate easily. System’s Usability needs to be a key component as it will reduce the number of change request from the user.

1. **Platform Used**

The Warehouse Management System (WMS) has been implemented using VB.NET programming language and developed using the Visual Studio IDE.

1. **Database Used**

WMS utilities the MS SQL Server Database for the storage of the Stocks, Stock Category etc.

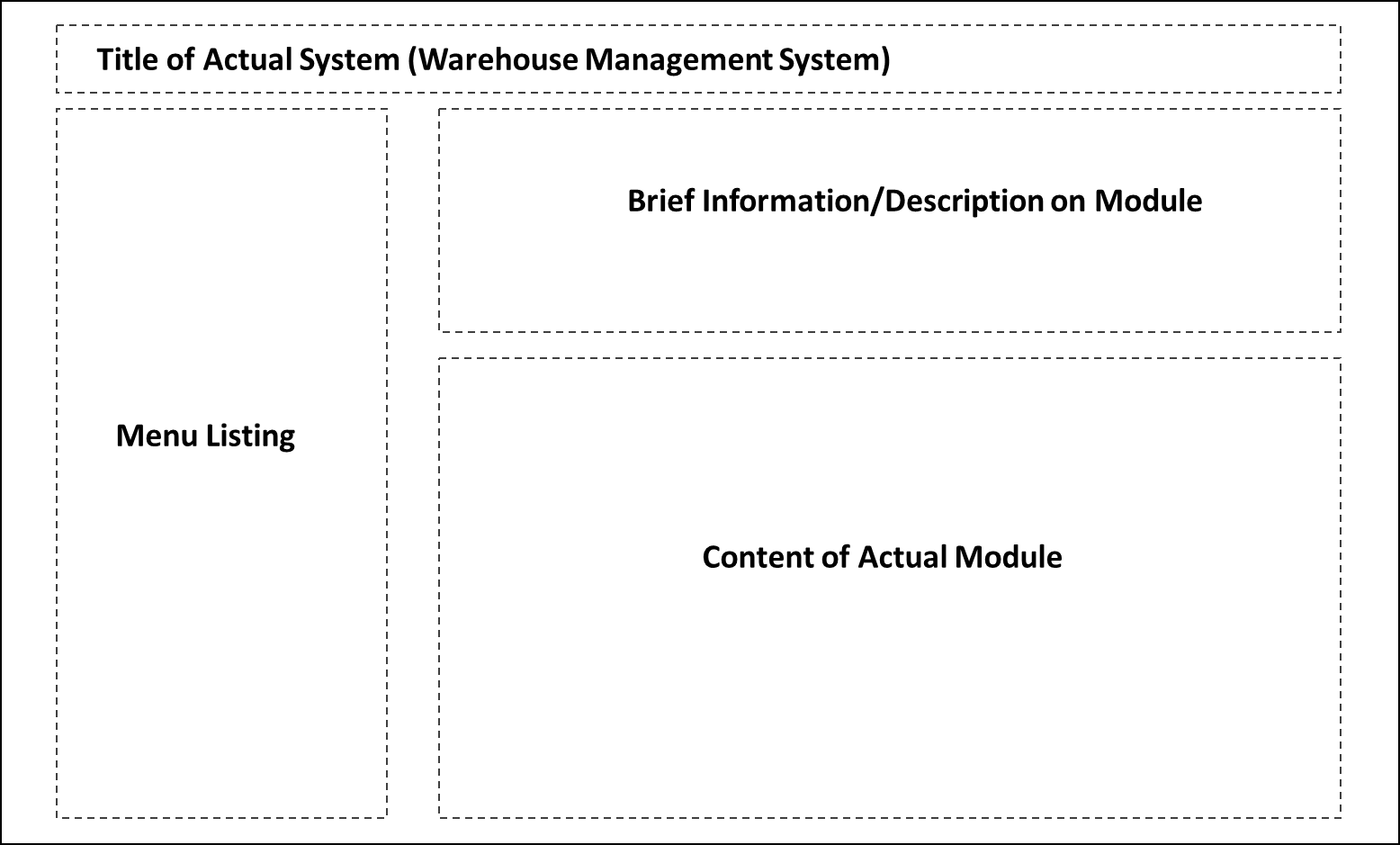
1. **GUI**

WMS is a website that runs on localhost. Users are presented with a login page and subsequently upon login displayed with menu options on the left hand side.

## User Interface Design

The system will have an overall design layout as per below where each section will have their corresponding information displayed. There will be a menu list on the left of the system and a brief information of the module function on right top and the actual module on bottom right.

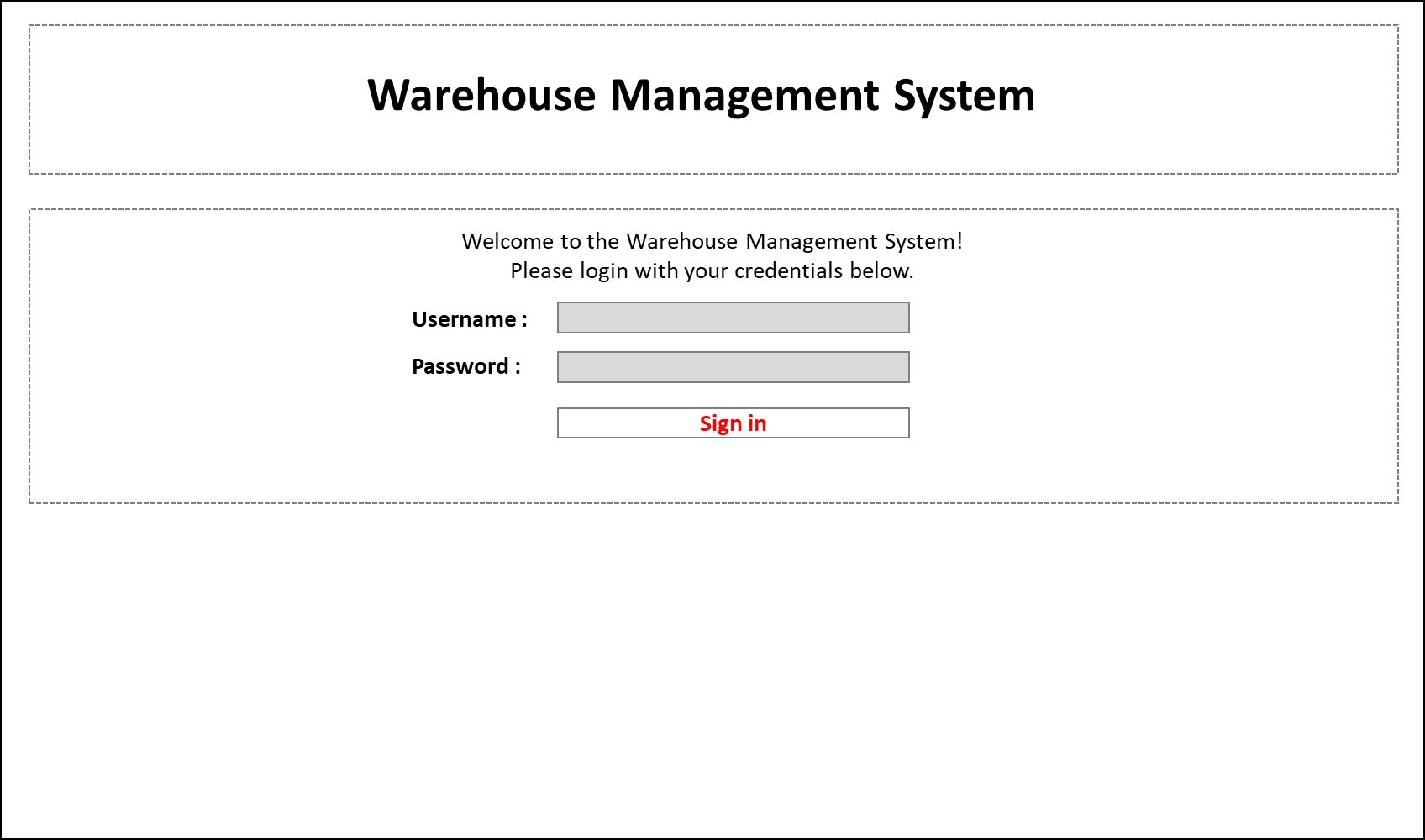
The main activity of the module will be at ‘Content of Actual Module’ section.

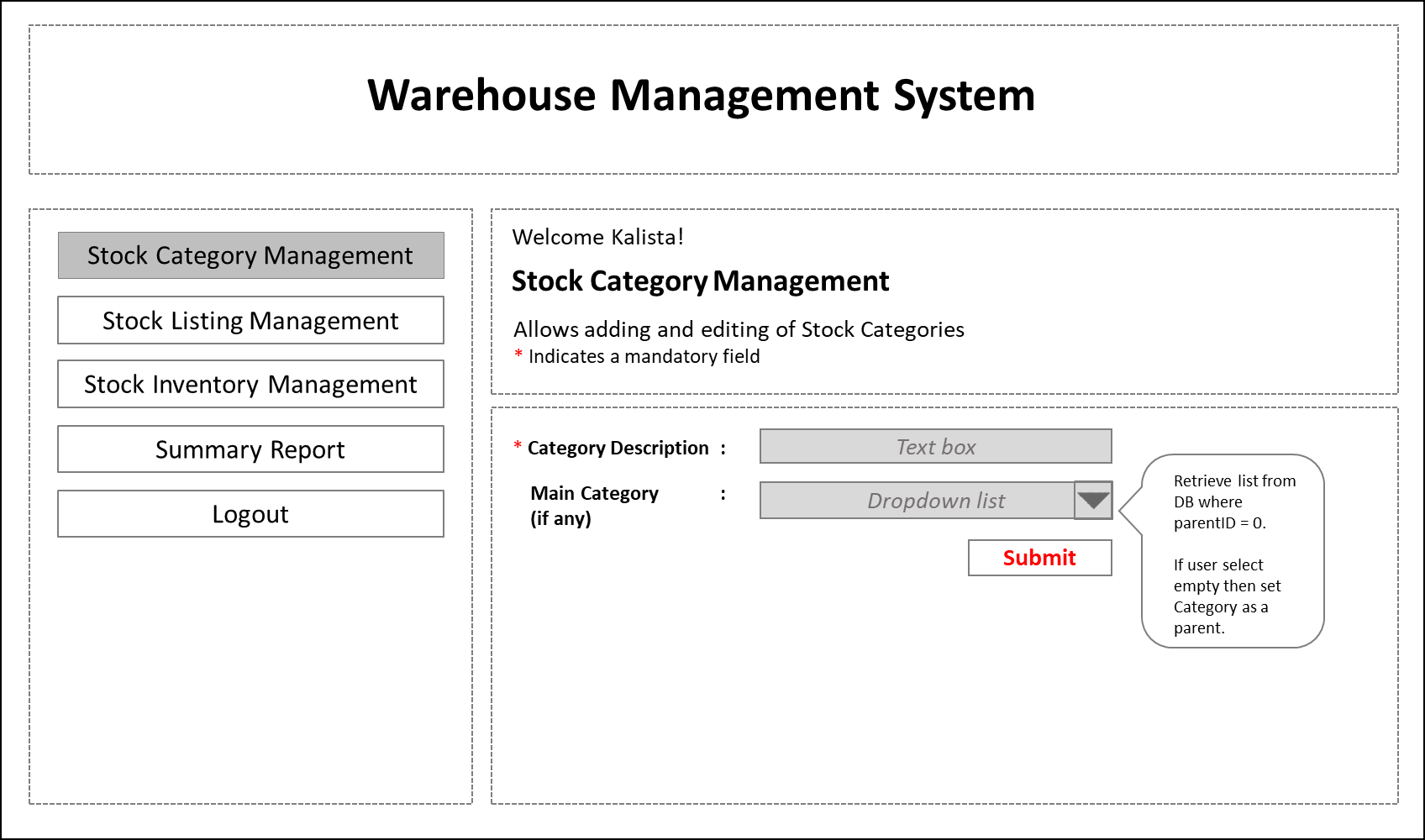


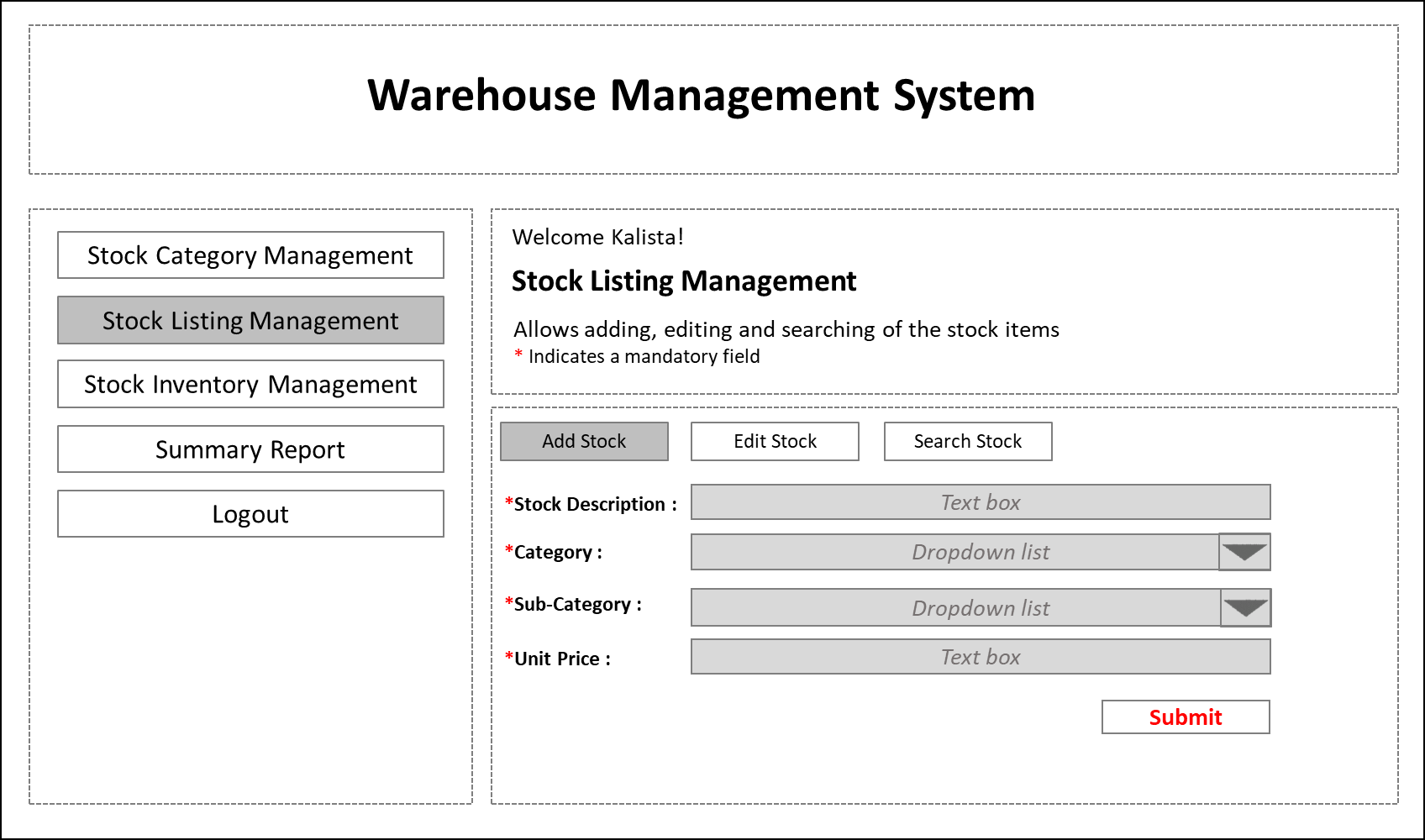
### Screen Designs (Prototype)

The screen designs below were done up as a rough guide before actual implementation.

Some examples below.







The entire Proposed Screen Flow in the below PowerPoint Slides.



## Functionalities of WMS

### Login

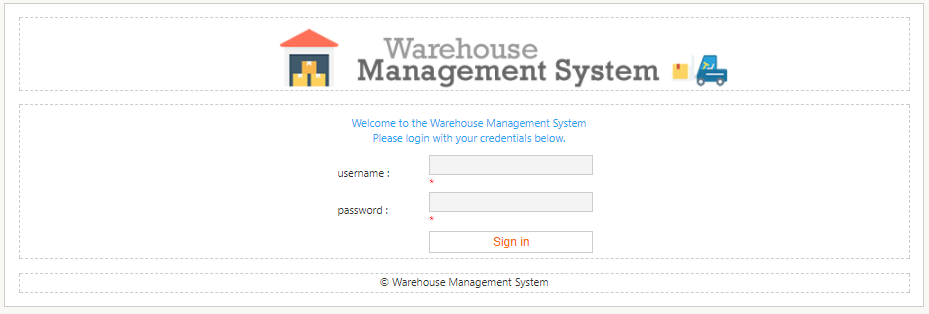
This module allows the user to login into WMS. System will also validate the encrypted password and will lock the account if the user has made 3 unsuccessful attempts, this will result in the user not being able to login once the 3 attempts have been used up.



* + - * Enter authentication credentials
      * Click on the **[Sign in]** button
      * System authenticates credentials and process accordingly
      * Account will be locked out after 3 unsuccessful attempts



System will validate the fields before processing the information. The mandatory field indicator will indicate any error **[\*]**



### Control Panel

This module is the first page that the user will see upon successful login. Stock Control Panel displays an overview of the stocks that are running low on stocks (quantity falling below the threshold).



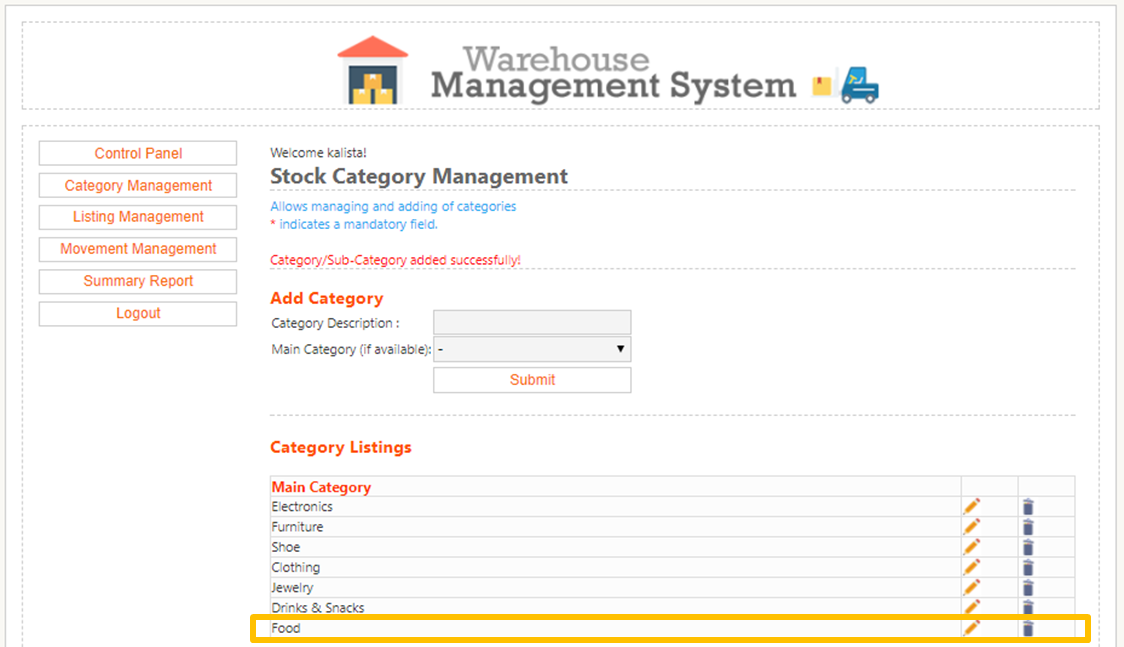
### Category Management

This module allows the user to manage the stock category of the system. The module will have a textbox to key in the category description and user will be able to select the parent category if any. If ‘-‘ is selected as the parent category then the category will be set as a parent category.

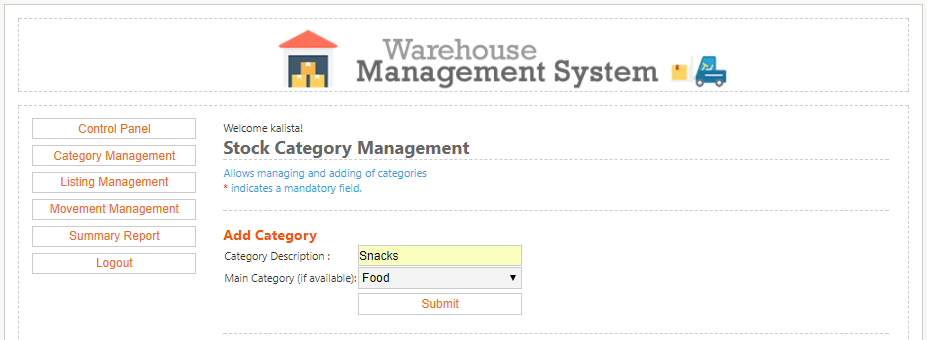
Adding Main Category



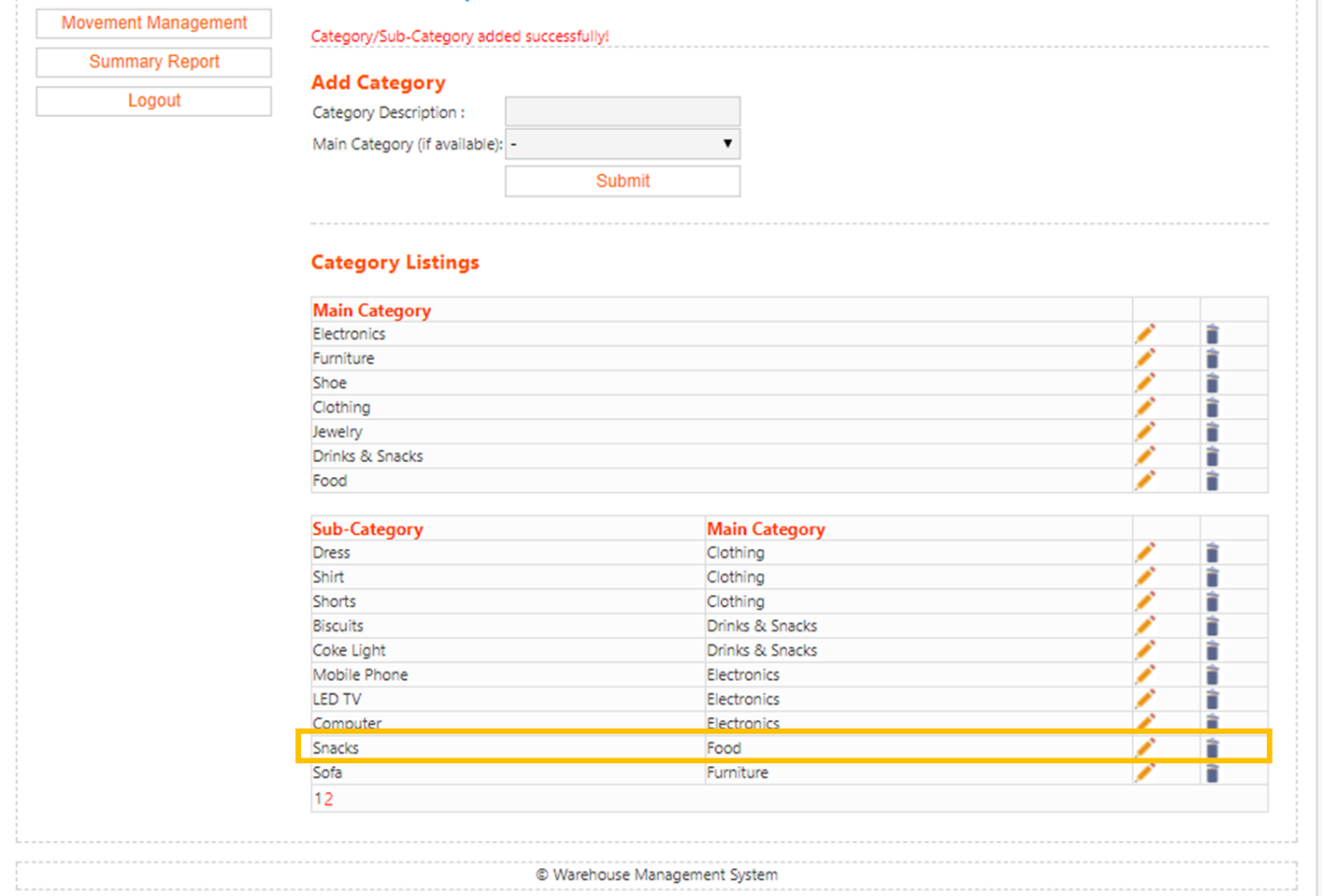
* + - * Enter Category Description
      * Select ‘-‘ as Main Category
      * Click on the **[Submit]** button
      * System captures Food as the Main Category and display in Main Category Listing table



Adding Sub-Category



* + - * Enter Category Description
      * Select ‘Food‘ as Main Category
      * Click on the **[Submit]** button
      * System captures Snacks as the Sub-Category and display in Sub-Category Listing table

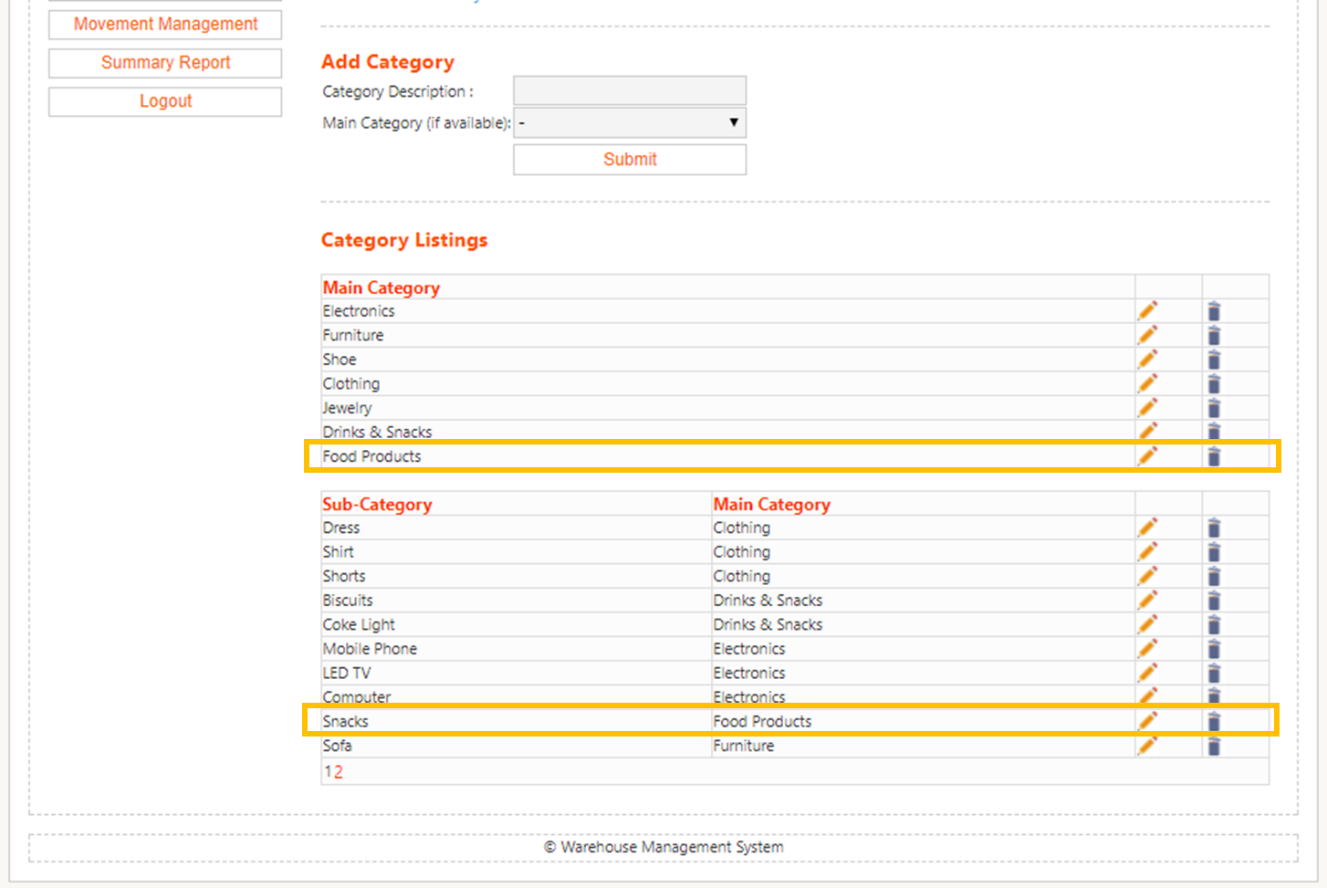


Edit Main Category

* + - * Click on the  icon

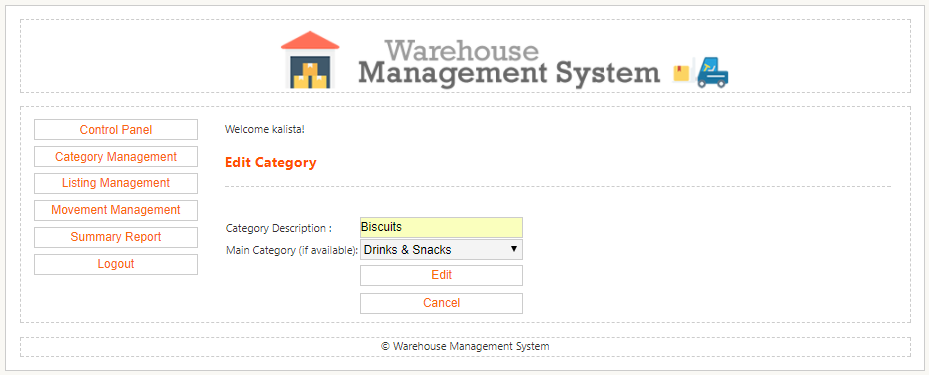


* + - * Edit Category Description
      * Click on the **[Edit]** button
      * System captures and updates both table listing from ‘Food’ to ‘Food Products’

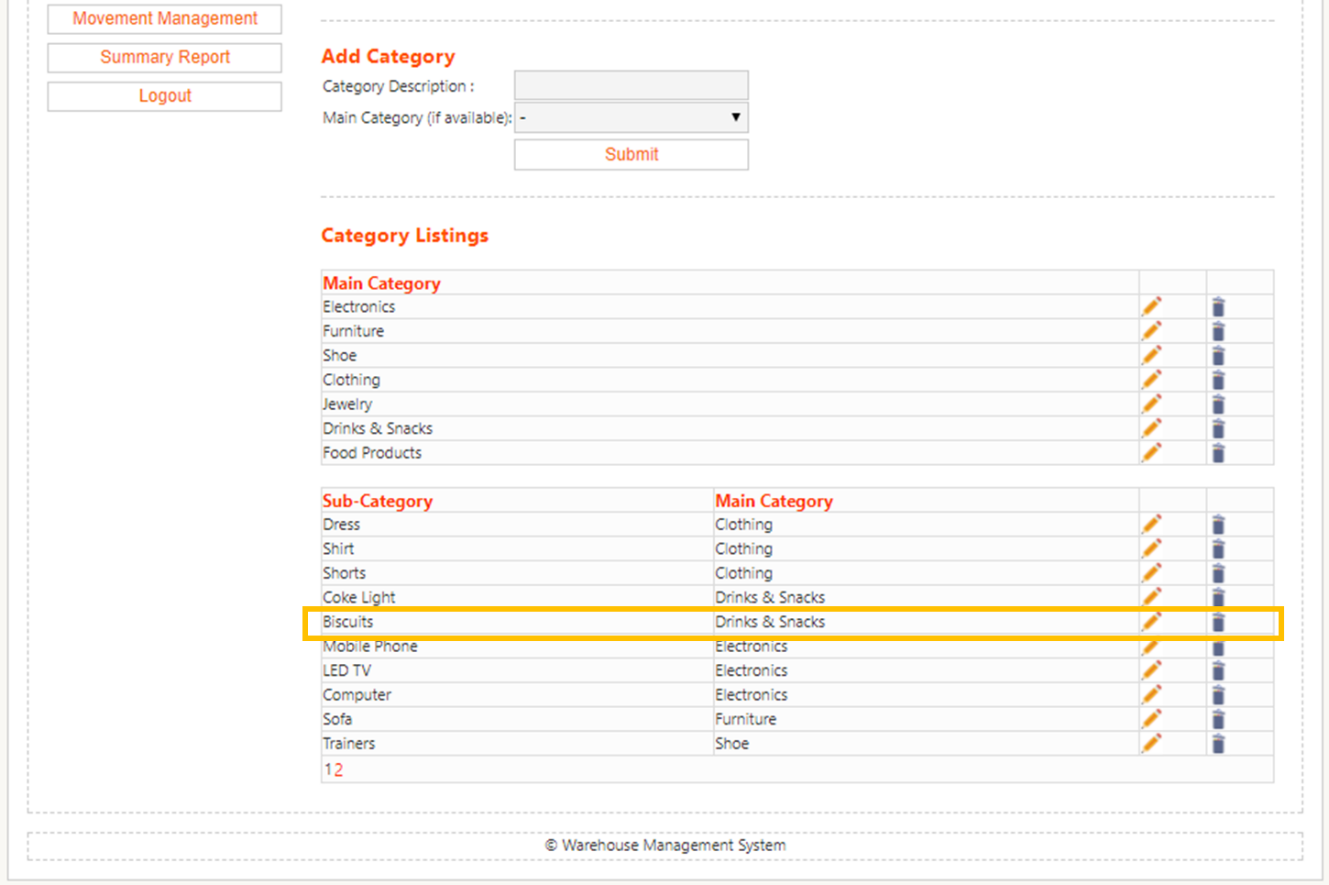


Edit Sub-Category

* + - * Click on the  icon

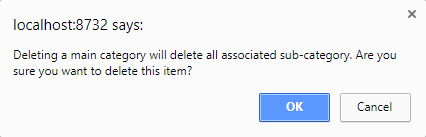


* + - * Edit Category Description
      * Edit Main Category
      * Click on the **[Edit]** button
      * System captures and updates Sub Category table listing

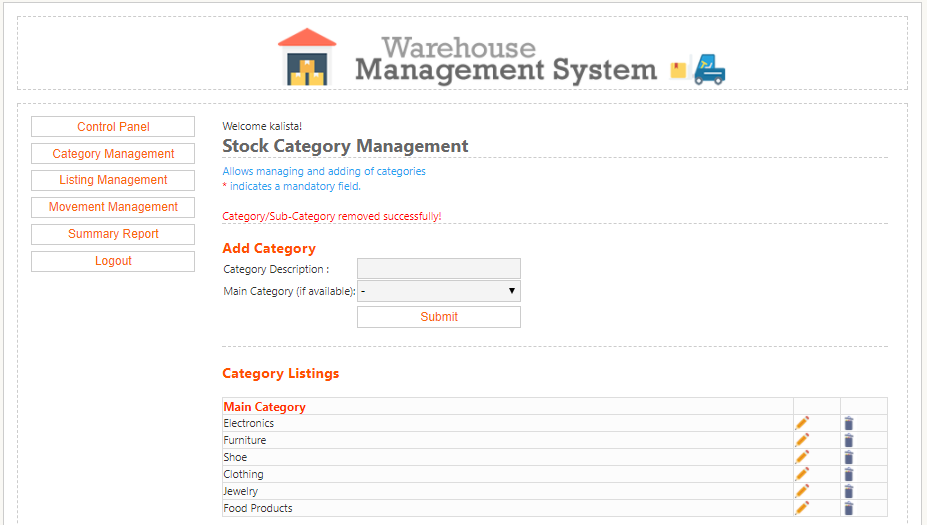


Delete Main Category

* + - * Click on the  icon

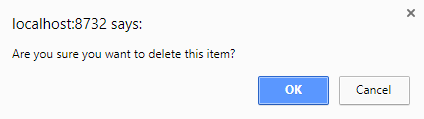


* + - * System prompts deletion confirmation
      * System displays ‘Category/Sub-Category removed successfully’ and removes from table listing (Drinks & Snacks removed from list)

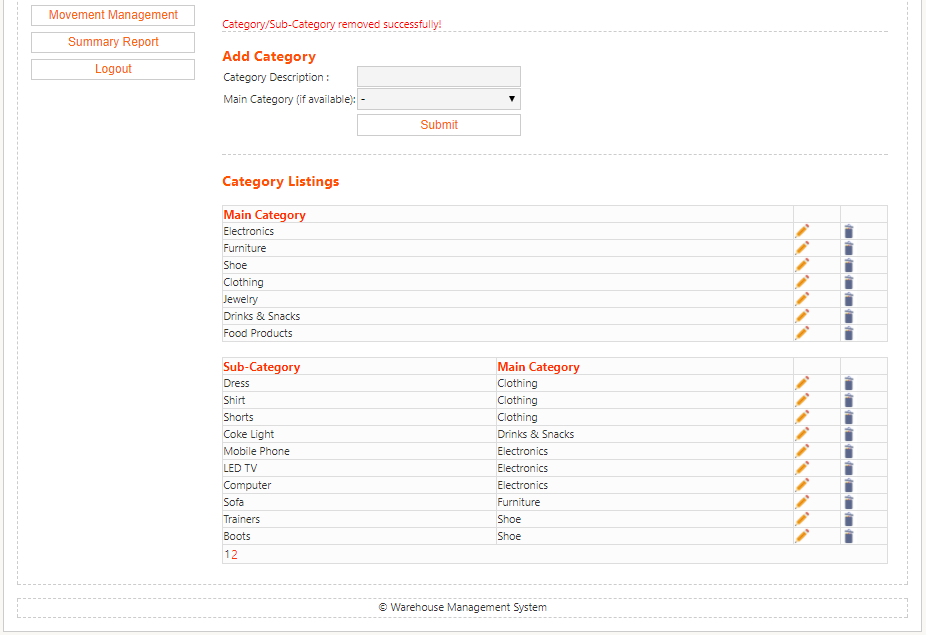


Delete Sub-Category

* + - * Click on the  icon



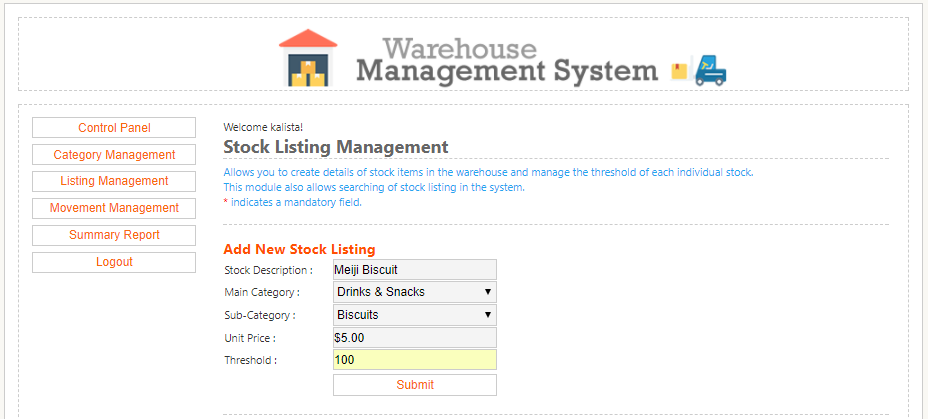
* + - * System prompts deletion confirmation
      * System displays ‘Category/Sub-Category removed successfully’ and removes from table listing (Main Category: Biscuits and Sub-Category: Drinks & Snacks removed)



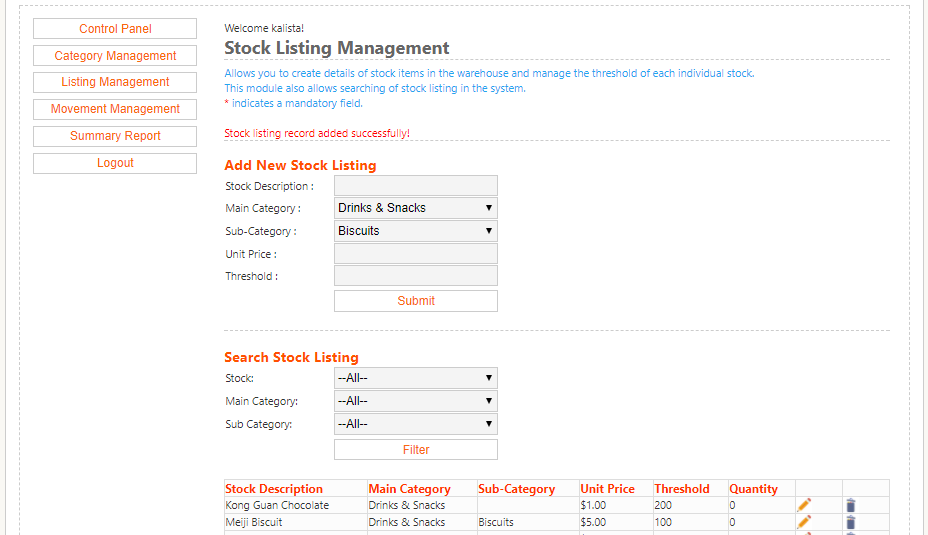
### Listing Management

This module allows the user to manage the different stock item that the warehouse holds. User is able to Add/Edit the stock item. User will have to create the stock first before any stock movement; stock in or stock out that can be perform.

Add Stock Listing



* + - * Enter New Stock Listing description
      * Click on **[Submit]** button
      * System displays ‘Stock listing record added successfully!’
      * Record added in table listing

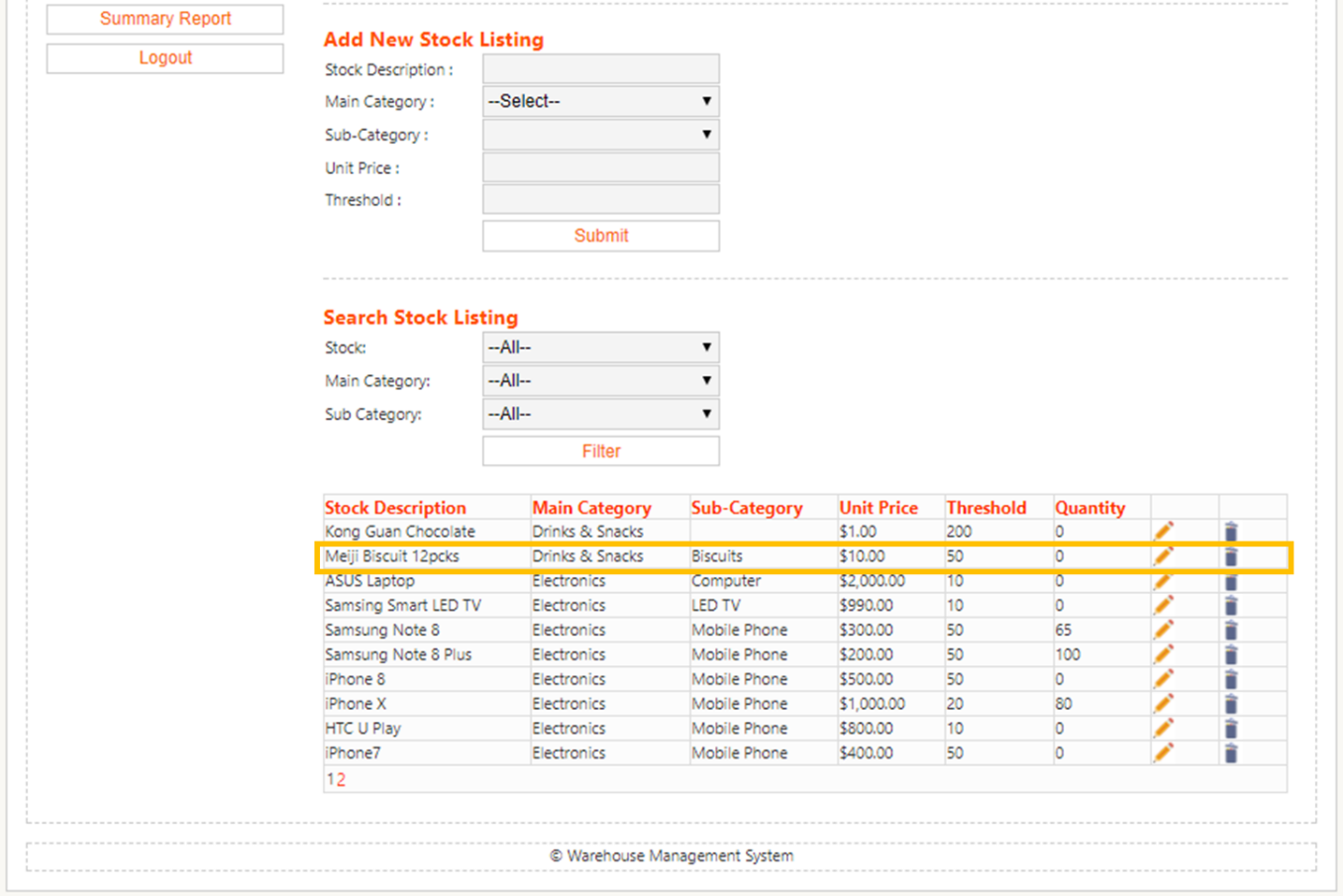


Edit Stock Listing

* + - * Click on the  icon

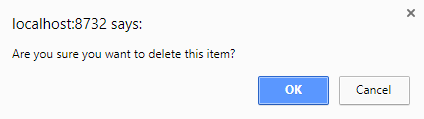


* + - * Enter the details
      * Click on **[Edit]** button
      * System displays ‘Stock listing record added successfully!’
      * Record added in table listing

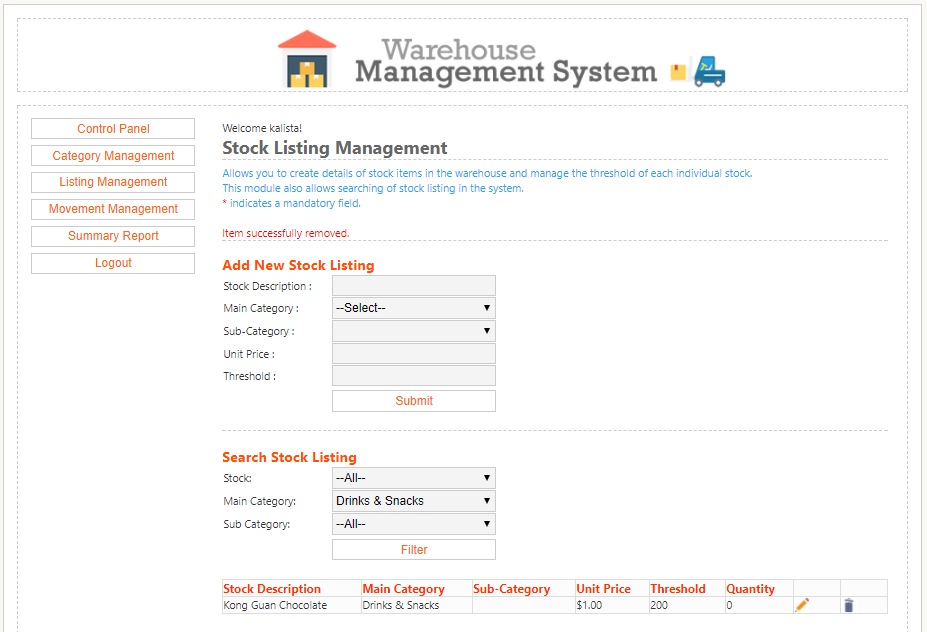


Delete Stock Listing

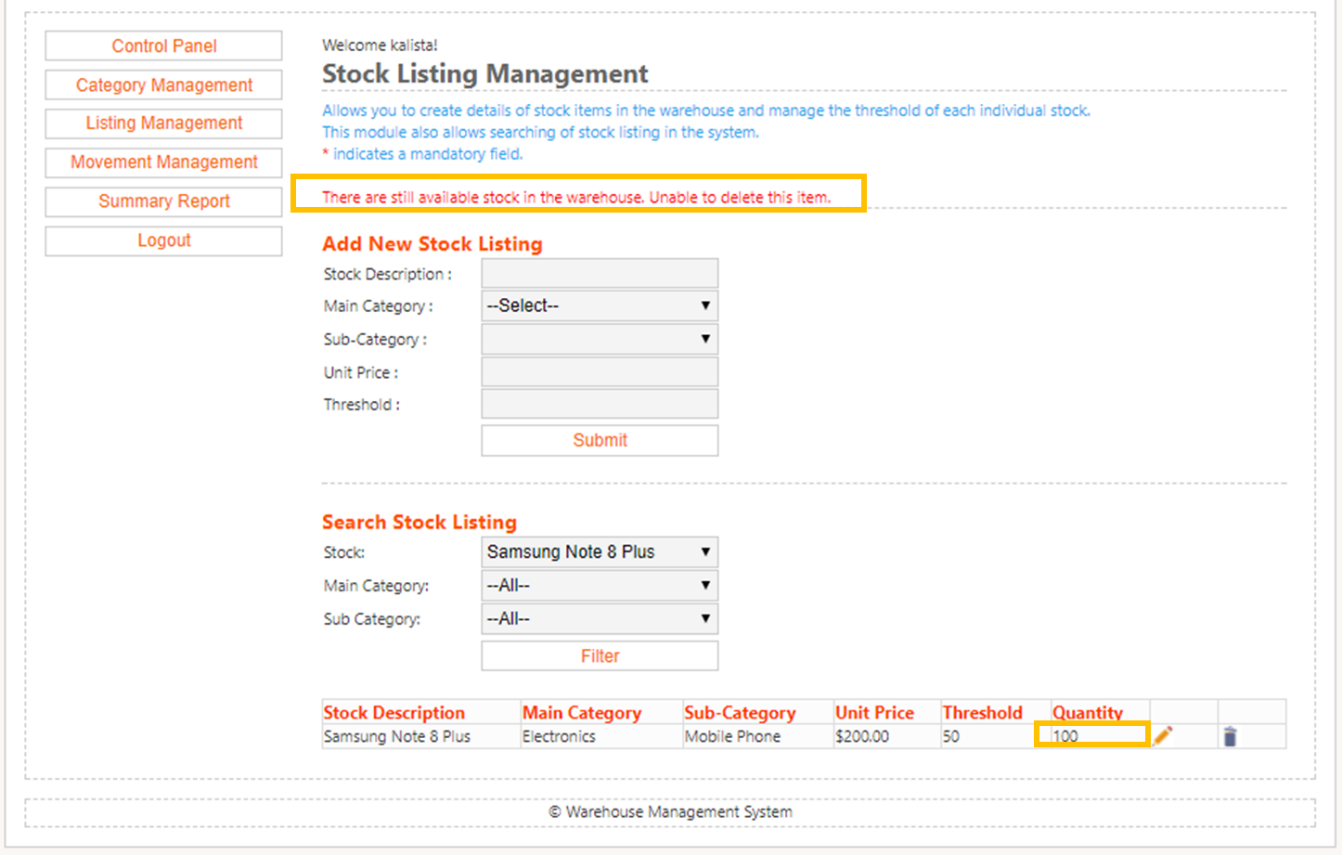
* + - * Click on the  icon



* + - * System displays ‘Item successfully removed.’
      * Item is removed from the table listing

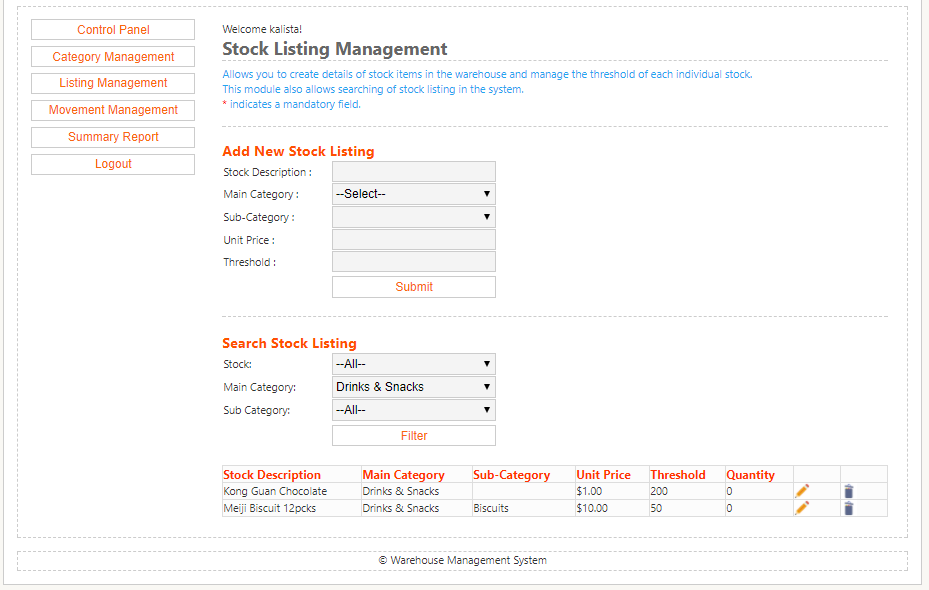


* + - * System will check if there are remaining stocks in system before removing
      * Item cannot be removed if stock quantity > 0



Search Stock Listing

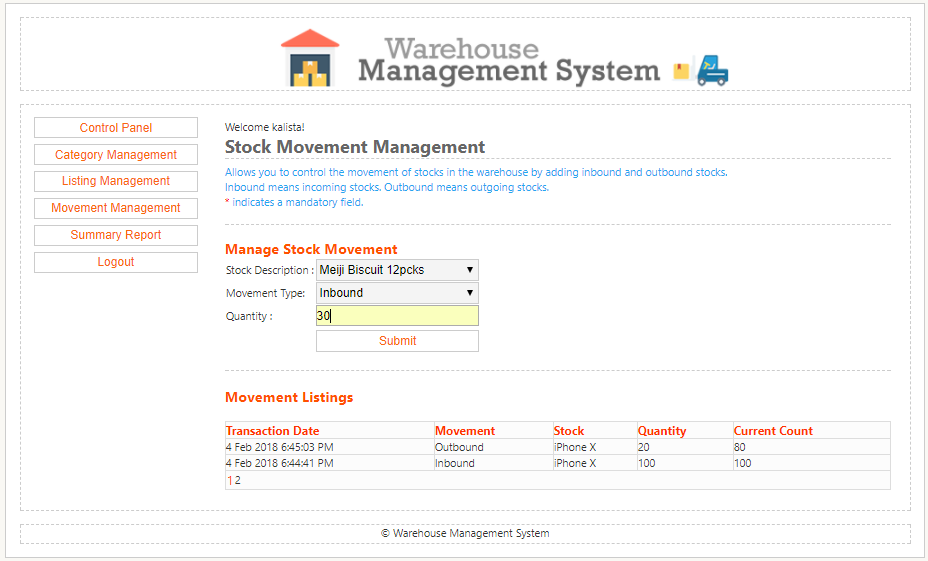
* + - * Enter the search details
      * Click on **[Filter]** button
      * System displays the results by the filter criteria



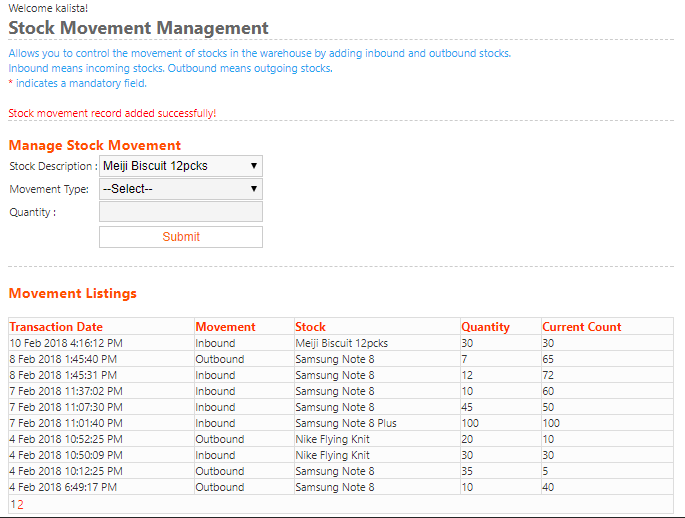
### Movement Management

This module allows the user to enter stock movement into the system to keep track of current stock level real time.

Add Inbound Stocks



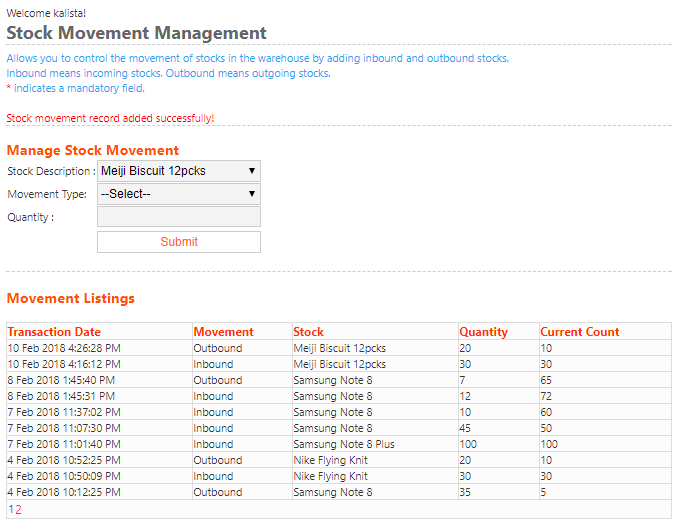
* + - * Pick from dropdown the Stock description
      * Pick from dropdown Movement type for Inbound
      * Enter Quantity
      * Click on **[Submit]** button
      * System adds the ‘Inbound’ record in the Movement Listings table



Add Outbound Stocks

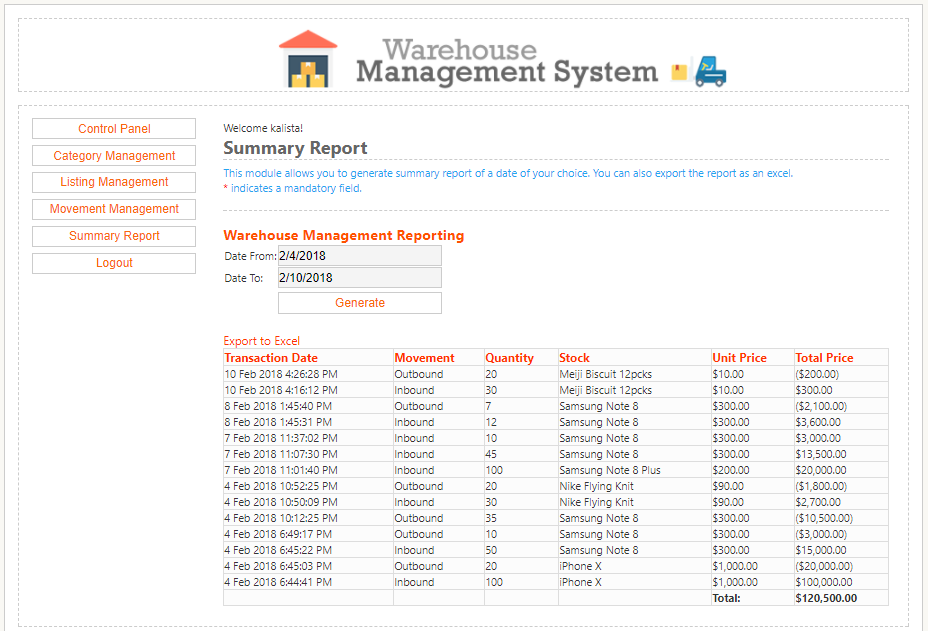


* + - * Pick from dropdown the Stock description
      * Pick from dropdown Movement type for Outbound
      * Enter Quantity
      * Click on **[Submit]** button
      * System adds the ‘Outbound’ record in the Movement Listings table

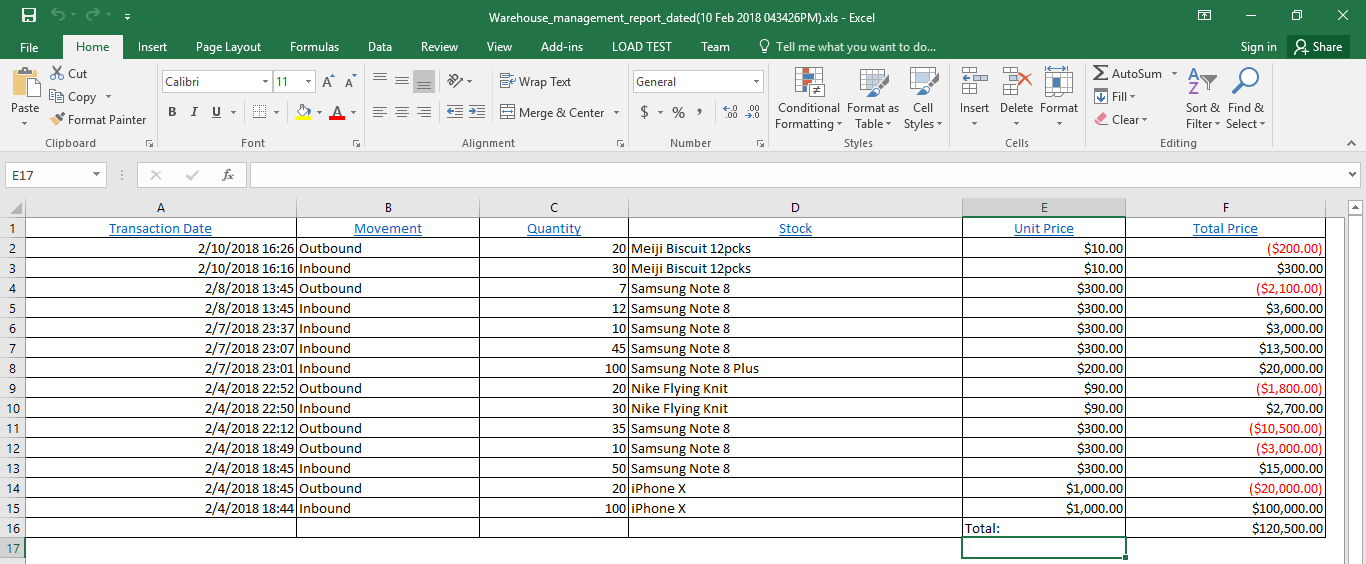


### Summary Report

This module allows the user to generate reports based on the Date From and Date to criteria. The report will display the stock movement transaction that was done between this dates.

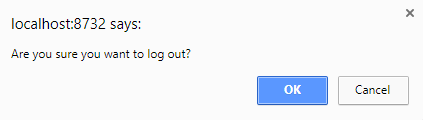


* + - * Pick from the date picker for Date From and Date To
      * Click on **[Generate]** button
      * System displays list of transactions between this dates
      * System allows export of report to excel



### Logout

This module allows the user to logout from the warehouse management system.



* + - * User click on Logout at Menu
      * System prompts logout confirmation
      * Click on **[OK]** button

## Use Case Diagram

### Iteration 1

A close up of a map

Description generated with high confidence

In the process of coding out the Warehouse Management (WM) Tool, our team have made adjustment

to the Use Case Diagram as shown above.

The primary actor “Staff” will be able login, manage stock category along with stock listing and stock

movement and last but not least, view summary report.

### Iteration 2

A picture containing text, map

Description generated with very high confidence

Halfway through the completion of the Warehouse Management (WM) Tool, we have remove

“Decrypt Password” which is initially included inside the Use Case “Login”. This is because upon

login successfully, the password that is keyed in will be harsh, followed by encryption and will

be sent to the database.

Under the Use Case “Manage Stock Category”, we have remove “Search Category” as the

Categories and Sub-Categories will be retrieved from the database and listed out orderly in a table

form upon accessing “Manage Stock Category” tab. **Final Version**

A picture containing text, map

Description generated with very high confidence

The above is our team’s finalise version of the Use Case Diagram whereby the Staff will only be able to gain access to the main function “Manage Stock Category”, “Manage Stock Listing”, “Manage Stock Movement” and “View Summary Report” after logging to the system.

## Use Case Detailed Description

### Detailed Use Case – Login

|  |  |
| --- | --- |
| **Use Case Description** | This use case describes how the Staff login into the system |
| **Primary Actor (s)** | Staff |
| **Pre-conditions** | User is required to have an account in the system |
| **Flow** | **Main Flow :**   1. System will display a page to allow user to key in his/her username and password 2. Staff will input the login credential and click on “Sign In” button 3. System will run hashing algorithm on the password inputted 4. System will verify if the hash inputted matches Staff’s hashed password 5. System then grants Staff the access and proceed to the main page 6. Use case ends   **Alternate Flow (AF1) – Invalid Username / Password**   1. Staff insert wrong username / password 2. System display error message “Authentication failed!” 3. Use case ends   **Alternate Flow (AF2) – 3 unsuccessful login attempts**   1. Staff input wrong username / password for more than 3 times 2. System will lock that particular user’s account 3. System display error message “Your account has been locked out!” 4. Use case ends |
| **Post-conditions** | 1. System will proceed to home page with access to various functions |
| **Business Rules** | 1. Mandatory fields must be entered. |

### Detailed Use Case – Manage Stock Category

|  |  |
| --- | --- |
| **Use Case Description** | This use case describes how a staff add, edit and delete Stock Category in the system |
| **Primary Actor (s)** | Staff |
| **Pre-conditions** | User must first login to the system |
| **Flow** | **Main Flow :**   1. Staff will login to the system with his/her username and password 2. Staff will click on the “Category Management” tab 3. System will display all the main category and sub-category retrieved from the database 4. Staff will key in the category description and select “-“ under the dropdown list (**if there isn’t the correct category available**) 5. Staff will select the main category under the dropdown list (**if there is any**) 6. System will store the information into the database 7. Use case ends   **Alternate Flow (AF1) – Edit Stock Category**   1. Staff will click on the pencil icon beside the main category title that he/she wants to edit 2. Staff will key in the correct category name 3. System will store the edited information into the database 4. Use case ends   **Alternate Flow (AF2) – Delete Stock Category**   1. Staff will click on the bin icon beside the main category title that he/she wants to delete 2. System will prompt the user “Are you sure you want to delete this item?” 3. Staff select “OK” 4. System will delete the stock category from the database 5. Use case ends |
| **Post-conditions** | 1. All the stock categories will be stored in the system |
| **Business Rules** | 1. Mandatory fields must be entered. |

### Detailed Use Case – Manage Stock Movement

|  |  |
| --- | --- |
| **Use Case Description** | This use case describes how a staff manages both the inbound and outbound stock movement in the system |
| **Primary Actor (s)** | Staff |
| **Pre-conditions** | User must first login to the system |
| **Flow** | **Main Flow :**   1. Staff will login to the system with his/her username and password 2. Staff will click on the “Movement Management” tab 3. System will display all the inbound and outbound stock’s details 4. Staff will key in the stock description, quantity and select “inbound” or “outbound” for the movement type 5. System will store the information into the database 6. Use case ends |
| **Post-conditions** | 1. All the stock movement added will be stored in the system |
| **Business Rules** | 1. Mandatory fields must be entered. |

### Detailed Use Case – Manage Stock Listing

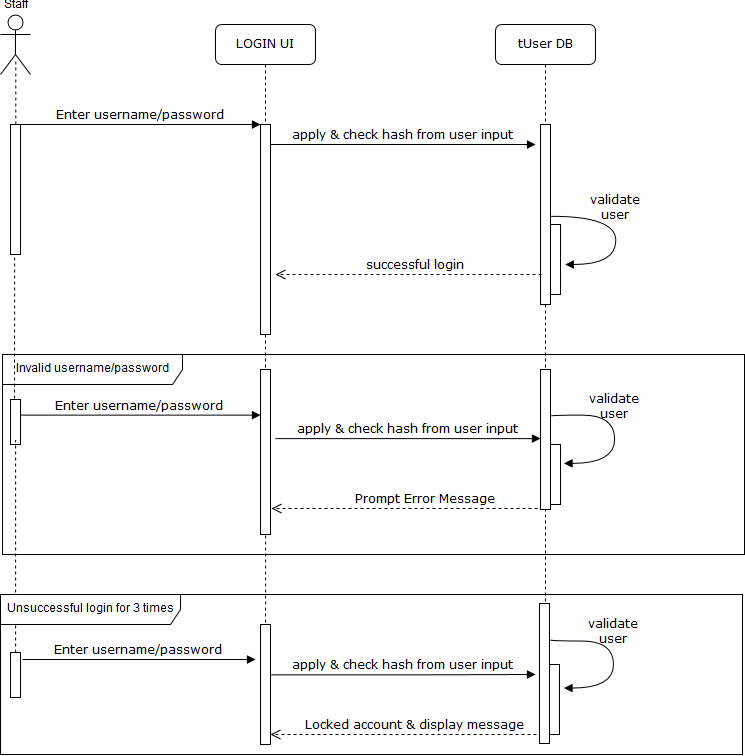
|  |  |
| --- | --- |
| **Use Case Description** | This use case describes how a staff manage stock listing in the system |
| **Primary Actor (s)** | Staff |
| **Pre-conditions** | User must first login to the system |
| **Flow** | **Main Flow :**   1. Staff will access the listing management tab 2. Staff will click on the “Add New Stock Listing” tab 3. Staff will enter new stock listing description and submit. 4. System will store the information and display ‘Stock listing record added successfully’. 5. System will display the added stock information 6. Use case ends   **Alternate Flow (AF1) – Edit stock listing**   1. Staff will click on the  button beside the stock that is going to be edited 2. Staff will edit the stock listing information 3. System will save the changes made and displays ‘Stock listing record added successfully!’ 4. System will display the edited stock listing 5. Use case ends   **Alternate Flow (AF2) – Delete stock listing**   1. Staff will click on the  button beside the stock that is going to be deleted. 2. System will prompt with a pop-up box to ask user for confirmation to permanently delete the stock listing. 3. User selects OK. 4. System will remove the stock and display ‘Item successfully removed.’ 5. Use case ends   **Alternate Flow (AF3) – Search stock listing**   1. Staff will access the listing management tab. 2. Staff will enter search details 3. Staff will click on the filter button 4. System display results via the filter criteria 5. Use case ends |
| **Post-conditions** | 1. All the stock listing added will be stored in the system and displayed to user. |
| **Business Rules** | 1. Mandatory fields must be entered. |

Detailed Use Case – View Summary Report

|  |  |
| --- | --- |
| **Use Case Description** | This use case creates summary report for the Staffs |
| **Primary Actor (s)** | Staff |
| **Pre-conditions** | User must first login to the system |
| **Flow** | **Main Flow**   1. Staff access the summary report tab. 2. Staff selects start date and end date of transactions for the report. 3. System displays list of transactions between these dates for viewing. 4. System allows export of report to excel. |
| **Post-conditions** | 1. Report from database is generated as excel. |
| **Business Rules** | 1. Mandatory fields must be entered |

## Sequence Diagram

### Sequence Diagram – Login



The Sequence Diagram above show the process of how the user will login to the system.

### Sequence Diagram – Manage Stock Category

A close up of a map

Description generated with very high confidence

The Sequence Diagram above show the process of how the user will add, edit and delete a Category / Sub-Category in the system.

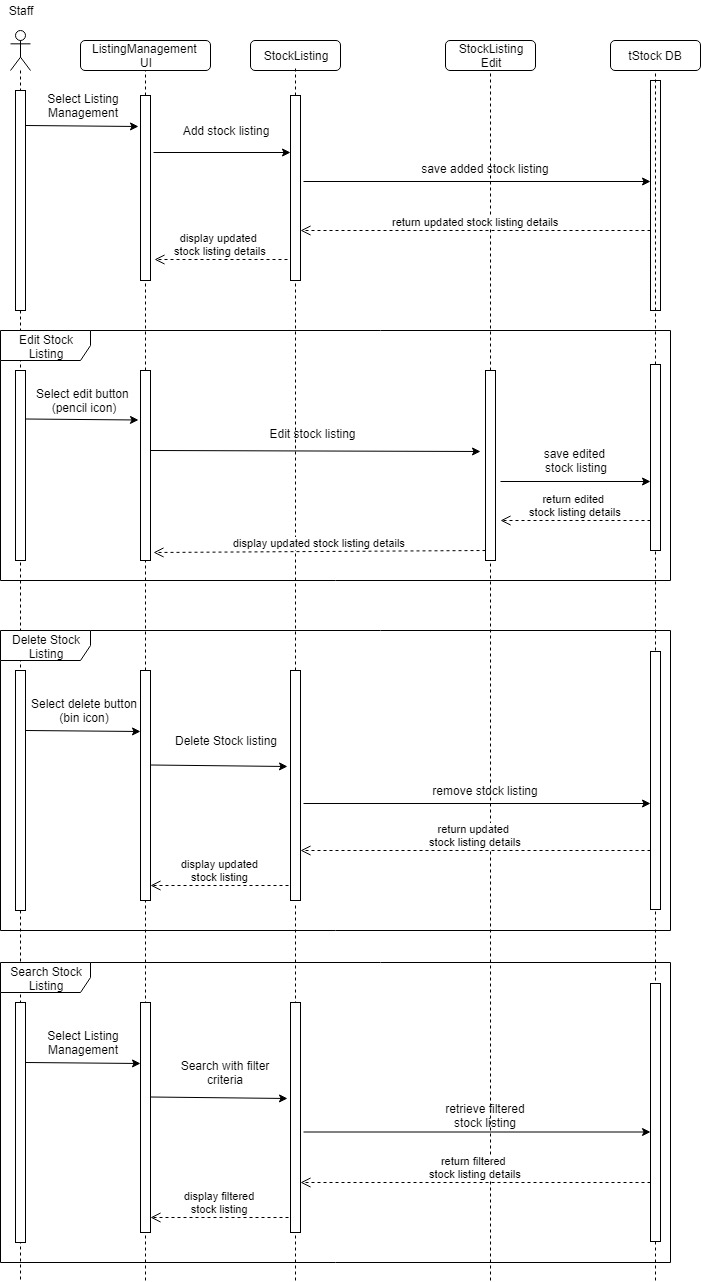
### Sequence Diagram – Manage Stock Movement

A screenshot of a cell phone

Description generated with high confidence

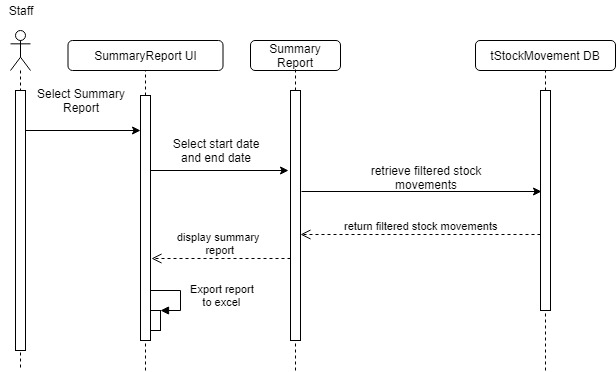
The Sequence Diagram above show the process of how the user will add a record of the Stock Movement in the system.

### Sequence Diagram – Manage Stock Listing



The Sequence Diagram above show the process of how the user will add, edit, delete and search a stock listing in the system.

### Sequence Diagram – View Summary Report

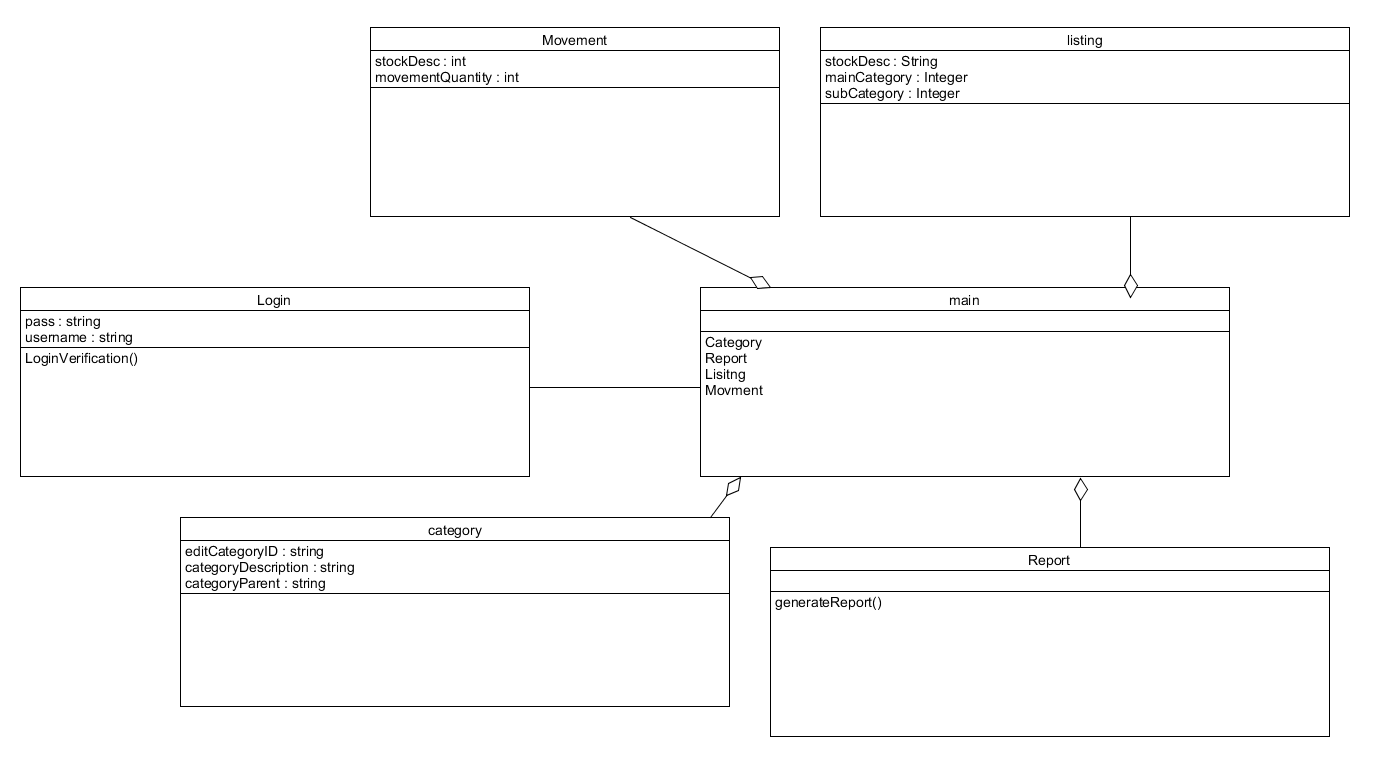


The Sequence Diagram above show the process of how the user can view and export the summary

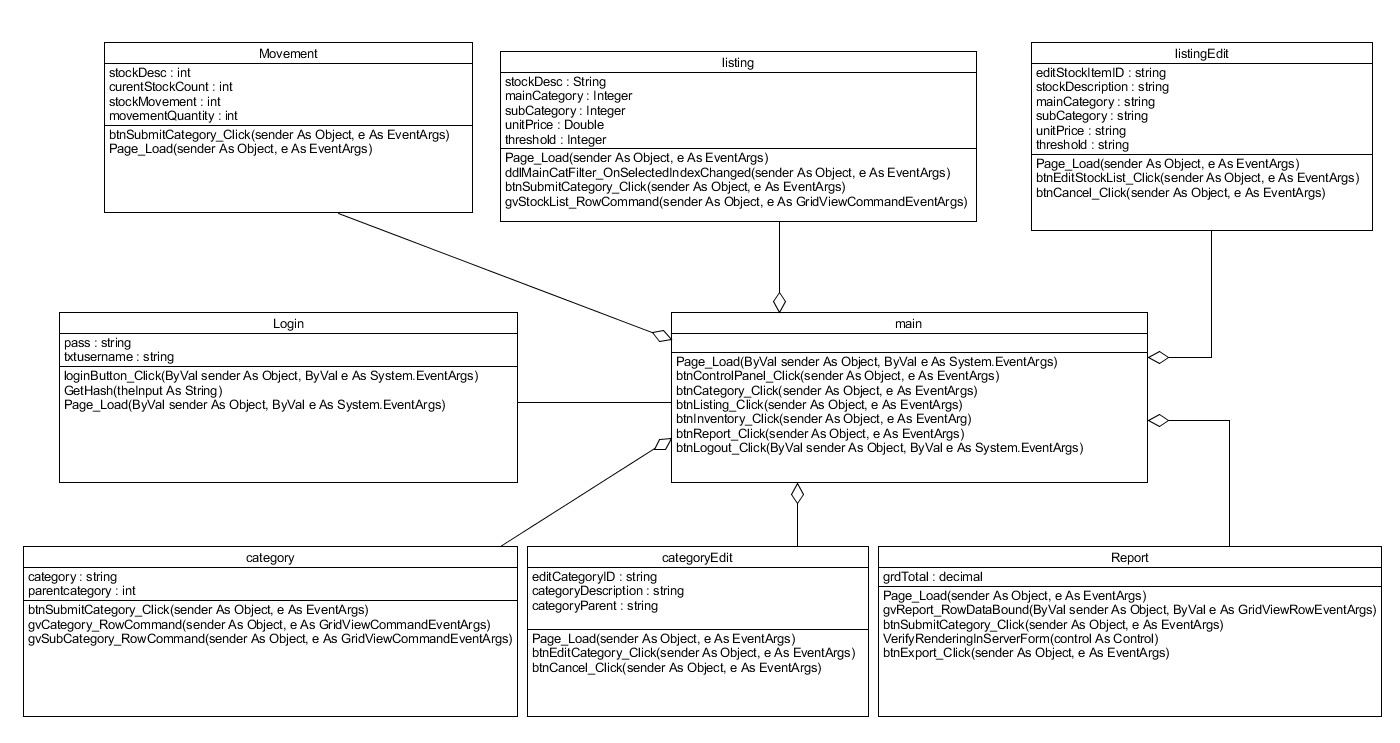
report.

## Class Diagram & Description

### Iteration 1



### Final Version



### Class Description

|  |  |
| --- | --- |
| Class Name: | Login |
| Html File: | login.aspx |
| Implementation file: | Login.apsx.vb |
| Description  Login.aspx is responsible for showing the Login UI for the user to input username and password.  Login.apsx.vb is responsible to hold the logic such as to verify user login credential. | |

|  |  |
| --- | --- |
| Class Name: | Main |
| Html File: | Main.master |
| Description  Main.master is responsible for showing the standard User Interface to be used across all the html file. | |

|  |  |
| --- | --- |
| Class Name: | Movement |
| Html File: | Movement.aspx |
| Implementation file: | Movement.apsx.vb |
| Description  Movement.aspx is responsible for showing the Movement management UI when the user is in the movement management webpage and it interacts with SQL to pull out data that the user has stored.  Movement.apsx.vb stores whatever that the user has input into the database.  Inbound means incoming stock and outbound means out going stock. | |

|  |  |
| --- | --- |
| Class Name: | Listing |
| Html File: | Listing.aspx |
| Implementation file: | Listing Listing.apsx.vb |
| Description  Listing.aspx is responsible for showing the Listing management UI when the user is in the Listing management webpage and it interacts with SQL to pull out whatever data that the user has stored.  Listing.apsx.vb is to store to data that the user has input. | |

|  |  |
| --- | --- |
| Class Name: | ListingEdit |
| Html File: | ListingEdit.aspx |
| Implementation file: | ListingEdit.apsx.vb |
| Description  ListingEdit.aspx is responsible for showing UI after user have click on an edit button (a pencil image) from Lisitng.aspx.  Listing.apsx.vb is to update the data of what the user has intend to change. | |

|  |  |
| --- | --- |
| Class Name: | Category |
| Html File: | login.aspx |
| Implementation file: | Login.apsx.vb |
| Description  Category.aspx is responsible for showing the Category management UI when the user is in the Category management webpage and it interacts with SQL to pull out whatever data that the user has stored.  Category.apsx.vb is to store to data that the user has input. | |

|  |  |
| --- | --- |
| Class Name: | CategoryEdit |
| Html File: | login.aspx |
| Implementation file: | Login.apsx.vb |
| Description  CategoryEdit.aspx is responsible for showing UI after user have click on an edit button (a pencil image) from Lisitng.aspx.  CategoryEdit.apsx.vb is to update the data of what the user has intend to change. | |

|  |  |
| --- | --- |
| Class Name: | Report |
| Html File: | report.aspx |
| Implementation file: | report.apsx.vb |
| Description  report.aspx is responsible for showing the UI when user is in the report summary webpage.  report.apsx.vb is responsible to hold the main logic of the report and interaction with the database to generate a report for user to view. | |

## Data Dictionary

Below table show the data dictionary for our system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **tCategory** |  |  |  |  |
| **Field Name** | **Data Type** | **Field size** | **Description** | **example** |
| categoryId | int |  | Unique identifier for each category | 3 |
| categoryDescription | Varchar | 50 | Description for the item | Mobile Phone |
| categoryParent | int |  | Description of category of the item(drop down list; however in the database it will show in terms of int instead of the name) | 3 |
| createBy | Varchar | 50 | Name of the user create by | kalista |
| createOn | DateTime |  | store base on System Date and timestamp when user created the object | 04-Feb-18 10:52:25 PM |
| modifiedBy | Varchar |  | Name of the user who modified the item | kalista |
| modifiedOn | DateTime |  | store base on System Date and timestamp when user modified the object | 04-Feb-18 10:52:25 PM |
|  |  |  |  |  |
| **tStock** |  |  |  |  |
| **Field Name** | **Data Type** | **Field size** | **Description** | **example** |
| StockId | int |  | Unique identifier for each stock | 1 |
| StockDescription | Varchar | 50 | Description for the stock | Samsung Note 8 Plus |
| mainCategory | int |  | show the main category of the stock | 1 |
| SubCategory | int |  | show the sub category of the stock | 2 |
| unitPrice | float | 35 | Unit price for the stock | 300 |
| threshold | int |  | Minimum Quantity of the stock | 100 |
| quantityCount | int |  | Quantity of the stock | 10 |
| createBy | Varchar | 50 | Name of the user create by | kalista |
| createOn | DateTime |  | store base on System Date and timestamp when user created the object | 04-Feb-18 10:52:25 PM |
| modifiedBy | Varchar | 50 | Name of the user who modified the item | kalista |
| modifiedOn | DateTime |  | store base on System Date and timestamp when user modified the object | 04-Feb-18 10:52:25 PM |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **tStockMovement** |  |  |  |  |
| **Field Name** | **Data Type** | **Field size** | **Description** | **example** |
| StockMovementId | int |  | Unique identifier for each stock movement | 2 |
| StockItemId | int |  | each stock item id | 2 |
| movementType | int |  | 1 refer inbound(in-coming stock) , 2 refer outbound(outgoing stock) | 1 or 2 |
| movementQuantity | int |  | Quantity of the stock movement | 19-02-1900 |
| currentCount | int |  | it will show the current stock quantity | 30 |
| createBy | Varchar | 50 | Name of the user create by | kalista |
| createOn | DateTime |  | store base on System Date and timestamp when user created the object | 04-Feb-18 10:52:25 PM |
| modifiedBy | Varchar | 50 | Name of the user who modified the item | kalista |
| modifiedOn | DateTime |  | store base on System Date and timestamp when user modified the object | 04-Feb-18 10:52:25 PM |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **tUser** |  |  |  |  |
| **Field Name** | **Data Type** | **Field size** | **Description** | **Example** |
| id | Int |  | Unique identifier for every user ID | 1 |
| userName | Varchar | 50 | User Name for login | kalista |
| password | Varchar | 50 | Password for login( it is hash inside database) | 5F4DCC3B5AA  765D61D8327D  EB882CF99 |
| logincount | Int |  | Use to Lock account that have value more than 3 | 0 |

## Test Cases & Report

### Test Cases



Login & Control Panel



Stock Category Management

Stock Listing Management



Stock Movement Management



Summary Report

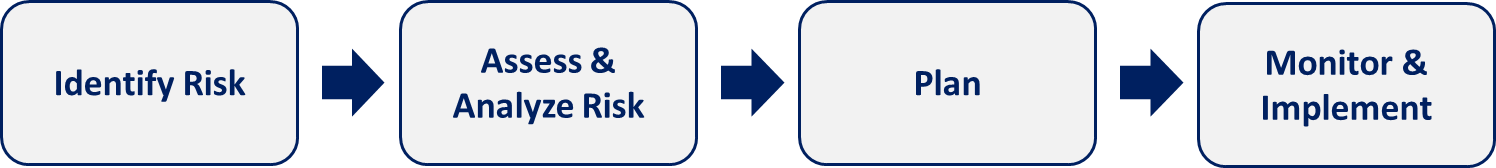


### Test Report



# Risk and Counter Measures

The following image below will explain how we are going to mitigate the risks in our project:



| **Risk** | **Likelihood** | **Impact** | **Mitigation Strategy** | **Resolution** |
| --- | --- | --- | --- | --- |
| Tight timeline (especially with the CNY period) | Medium | High | 1. Accommodate and find time after classes to meet up and discuss | 1. We have resolved this by meeting up after classes and during days that we do not have lessons to complete this assignment. We also used other forms of communication (i.e. WhatsApp) to catch up on our individual progress. |
| Unable to meet requirements using C++ due to lack of C++ programming languages. | High | High | 1. Use another coding language that we are familiar with and has a database structure that we are familiar with to meet with our system requirement. | 1. We have resolved this by deciding on using vb.net and visual studio IDE with SQL. |
| Problems installing Visual Studio IDE | High | High | 1. Implementer (Kalista) to help team mates in the installation 2. Do up screenshots of the systems in order to not waste any time | 1. We have resolved this by using team viewer to demonstrate the system to 1 another 2. Kalista did a demo of the entire system to the whole team during meet-ups 3. Screenshots/User Manual done up for the whole system |
| System defects found nearing to deadline | High | High | 1. Implementer (Kalista) to be on stand by and do a fix to the system and tester (Allen) to be on standby for testing | 1. Tester (Allen) had done a continuous testing for each module during the implementation phase to prevent any large impact defects found last minute |

# Meeting Minutes

**Meeting Minutes #1**

|  |  |  |
| --- | --- | --- |
|  | CSCI222- Systems Development  Assignment 1 | |
| Meeting Agenda & Minutes of Meeting | Page 1 of 2 |

| Meeting: | Assignment 2 – Meeting #1 | | |
| --- | --- | --- | --- |
| Date of Meeting:  (DD-MMM-YYYY) | *06-FEB-2018* | ***Time:*** | *08:30pm – 10:30pm* |
| Location: | Level 5 seating area | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **1. Meeting Objective** | | | |
| * Project Overview * Timeline * Task Allocation * Discussion on Architecure Design | | | |
| **2. Attendees** | | | |
| Name | Student Number | E-mail | |
| Kalista Chan | 5986345 | xkchan002@mymail.sim.edu.sg | |
| Teo Heong Hwee (Gavin) | 6098058 | hhteo005@mymail.sim.edu.sg | |
| Tam He Sheng (Allen) | 5986485 | hstam001@mymail.sim.edu.sg | |
| Jonas Chok | 6098459 | jwjchok001@mymail.sim.edu.sg | |
| Kyaw Myo Aung , Johns | 6097868 | Myoak001@mymail.sim.edu.sg | |
| **3. Meeting Agenda** | | | |
| **Topic** | | | **Owner** |
| Use Case Diagram | | | Gavin |
| Use Case Description | | | Gavin, Jonas & Johns |
| Sequence Diagram | | | Gavin, Jonas & Johns |
| Class Diagram | | | Allen & Kalista |
| Project Detailed Plan | | | Johns |
| Data Dictionary | | | Allen and Kalista |
| UI Design | | | Kalista |
| State Diagram | | | Johns & Jonas |
| **4. Pre-work/Preparation (documents/handouts to bring, reading material, etc.)** | | | |
| **Description** | | | **Prepared by** |
| UI workflow | | | Kalista |

| **5. Meeting Minutes** | | |
| --- | --- | --- |
| **Topic** | **Owner** | **Type** |
| **Meeting Agenda**   * Allen have allocate different role and assignment for each individual team member   **Use Case Diagram**   * Heong Hwee to design the Use Case Diagram   **Use Case Detailed**   * Heong Hwee, Jonas and Johns to write out Detailed Use Case   **Class Diagram & Data Dictionary**   * Allen and Kalista to discuss and design for class diagram & data dictionary   **Sequence Diagram**   * Gavin, Jonas and Johns to design the sequence diagram based on the used case diagram.   **Project Plan & State Diagram**   * Johns to design the state diagram & prepare the project timelime   **Meeting Minutes**   * Johns to write out the meeting minutes for team 1st meeting | All  Heong Hwee  Heong Hwee, Jonas, Johns  Allen & Kalista  Heong Hwee, Jonas, Johns  Jonas  Johns |  |

**Meeting Minutes #2**

|  |  |  |
| --- | --- | --- |
|  | CSCI222- Systems Development  Assignment 1 | |
| Meeting Agenda & Minutes of Meeting | Page 1 of 2 |

| Meeting: | Assignment 2 – Meeting #2 | | |
| --- | --- | --- | --- |
| Date of Meeting:  (DD-MMM-YYYY) | *14-FEB-2018* | ***Time:*** | *09:30pm – 10:30pm* |
| Location: | Level 1 seating area | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **1. Meeting Objective** | | | |
| * Sequence Diagram review * Use Case Daigram Review * Use Case Description Discussion * Overall report formatting | | | |
| **2. Attendees** | | | |
| Name | Student Number | E-mail | |
| Kalista Chan | 5986345 | xkchan002@mymail.sim.edu.sg | |
| Teo Heong Hwee (Gavin) | 6098058 | hhteo005@mymail.sim.edu.sg | |
| Tam He Sheng (Allen) | 5986485 | hstam001@mymail.sim.edu.sg | |
| Jonas Chok | 6098459 | jwjchok001@mymail.sim.edu.sg | |
| Kyaw Myo Aung , Johns | 6097868 | Myoak001@mymail.sim.edu.sg | |
| **3. Meeting Agenda** | | | |
| **Topic** | | | **Owner** |
| Use Case Diagram | | | Gavin |
| Use Case Description | | | Gavin, Jonas & Johns |
| Sequence Diagram | | | Gavin, Jonas & Johns |
| Class Diagram | | | Allen & Kalista |
| Project Detailed Plan | | | Johns |
| Data Dictionary | | | Allen and Kalista |
| UI Design | | | Kalista |
|  | | |  |
| **4. Pre-work/Preparation (documents/handouts to bring, reading material, etc.)** | | | |
| **Description** | | | **Prepared by** |
| UI workflow | | | Kalista |

| **5. Meeting Minutes** | | |
| --- | --- | --- |
| **Topic** | **Owner** | **Type** |
| **Meeting Agenda**   * Allen & Kalista to finalized the class diagram * Kalista to demo current progress of system finish up the coding/system * Gavin , Jonas & Johns to finalized the sequence diagram , usecase diagram & use case description   **System Update/Demo**   * Kalista update and demo the System on her laptop and get feedbacks from team-mates   **Use Case Diagram Update**   * Heong Hwee to design the Use Case Diagram   **Use Case Description Review and changes**   * Heong Hwee, Jonas update and detail checked on Use Case Description   **Class Diagram & Data Dictionary**   * Allen and Kalista to discuss and design for class diagram & data dictionary   **Sequence Diagram**   * Johns, Heong Hwee to finalized the sequence diagram to align with class diagram & used case diagrams.   **Meeting Minutes**   * Johns to write out the meeting minutes for team 2nd meeting | All  Kalista  Heong Hwee  Heong Hwee, Jonas  Allen & Kalista  Johns, Heong Hwee  Johns |  |