A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Title	:	**Object Oriented Programming Group Assignment Documentation**
Intake Code	:	**UCDF2309ICT(SE)**
Module Code	:	**AAPP013-4-2-OOP**
Level of Study	:	**Diploma Semester 4**
Module Lecturer	:	**Mr. Usman Hashmi**
Hand Out Date	:	**6 February 2025**
Submission Date	:	**24 March 2025**

Group Members :

Member	Name	TP Number
1	Paureen Tan Nie Nie	TP 075914
2	Lim Chee Xuan	TP 075916

Table of Contents

1.0 Introduction 4

1.1 System Overview 4

1.2 Assumptions 4

2.0 System Design and Implementation 5

2.1 Object-Oriented Programming (OOP) Concepts 5

2.1.1 Class 5

2.1.2 Object 6

2.2 Object-Oriented Programming (OOP) Principles 6

2.2.1 Encapsulation 6

2.2.2 Abstraction 7

2.2.3 Inheritance 8

2.2.4 Polymorphism 9

3.0 User Manual (GUI) 10

3.1 Login 10

3.2 Administrator 11

3.2.1 User and Entity Management (UEM) 11

3.2.2 Inventory Management 15

3.2.3 Transaction Records 18

3.2.4 Tracking and Reports 20

3.2.5 Log Activities 21

3.3 Staff 22

3.3.2 Modify Personal Details 22

3.3.3 Manage Inventory 23

3.3.4 Track Inventory 25

3.3.5 View Transactions 26

3.4 Logout 28

4.0 Conclusion 28

5.0 References 29

1.0 Introduction

The Personal Protective Equipment (PPE) Inventory Management System is designed to manage and track the inventory of PPE items for their stock levels and transaction records. This system allows administrators and staff to handle PPE inventory efficiently by performing main functionalities such as user management, inventory management, transaction tracking, and generating reports. This system is used to ensure that PPE stock is recorded correctly, monitor the stock levels to prevent low stock, and distribute it to hospitals.

1.1 System Overview

The system consists of two user types, which are administrator and staff. For administrators, they have full access to the system, where they can manage users, suppliers, hospitals, and PPE inventory. They can also track stock levels of the PPE inventory and generate reports of PPE item transaction records. For staff, they have limited access, where they mainly focus on updating stock when receiving PPE items from suppliers and distributing them to hospitals.

1.2 Assumptions

To ensure the system operates smoothly, there are some assumptions made:

- Each PPE item can only be supplied by one supplier, but one supplier can provide different types of PPE items.
- Users who enter incorrect login credentials more than three times will be automatically exited from the system.
- There is validation for password, email address and contact number, ensuring accuracy in data.
- Staff can only modify their own personal details, whereas the administrator has full control, including the ability to manage all user, such as administrator, staff, supplier, and hospital.
- The initial PPE item setup is done only once during the first execution of the system. After that, the system reads from the existing ppe.txt file.
- All PPE stock levels are recorded in boxes as the unit of measurement.

- Each transaction record is stored with a timestamp to track when the PPE item was received from a certain supplier or distributed to a certain hospital.
- All data is stored and retrieved from text files.
- Reports generated by the system are exported in PDF format.
- The system includes a log activity file for administrator to track key user actions such as login attempts, managing user and inventory updates.

2.0 System Design and Implementation

In this PPE Inventory Management System, multiple packages are structured, ensuring modularity and maintainability. Object-Oriented Programming (OOP) principles are applied in each class, enhancing the organization and reusability of the code.

2.1 Object-Oriented Programming (OOP) Concepts

2.1.1 Class

A class in Java is a blueprint for creating objects which have common attribute and behavior. It defines class variables and methods that objects will have. A class can only be declared once.

```java
package edu.oopgroup2.ppe.model.staffadmin;

import edu.oopgroup2.ppe.model.Entity;

public class User extends Entity {
    private String password, userType;

    //constructor
    public User() {}

    public User(String userID, String userName, String password, String userType, String userEmail, String userContact, String status) {
        super(userID, userName, userEmail, userContact, status);
        this.password = password;
        this.userType = userType;
    }

    //getters and setters
```

Figure 2.1.1 User Class

The `User` class is used to represent the administrator and staff users. It includes private attributes (`private String password, userType`), allowing them to only accessible within the certain class. Methods in this class are getters and setters, allowing controlled access or modify to these attributes when the other class calls the method.

2.1.2 Object

An object is an instance of a class. When created, each object has its own values for the attributes and behaviors defined in the class. A class can consist of multiple objects and objects can be created multiple times.

```
User newUser = new User(newUserID, userName, password, userType, userEmail, userContact, userStatus);
```

Figure 2.1.2 User Object

`newUser` object is created from the `User` class. This object stores unique details of staff/ administrator such as user ID, name, password, role, email, contact and status.

2.2 Object-Oriented Programming (OOP) Principles

2.2.1 Encapsulation

Encapsulation is the practice to keep class attributes private and providing public methods to access these private attributes.

```
private static final String USER_FILE_PATH = "txtFiles/users.txt";
```

Figure 2.2.1.1 Private Attribute

- `private` file path: Prevents external modification to the file path.

```
public String getPassword() {
    return password;
}
```

Figure 2.2.1.2 Public Method

- In the `User` class, `password` is declared as `private`, which does not allow direct access to them from outside the class.
- Therefore, `public` methods, which are getters and setters are used to access the private `password` attribute.

2.2.2 Abstraction

Abstraction in Java is used to hide complex internal implementation and show only important and relevant functionalities to the user.

In Java GUI, users interact with buttons and fields, but the backend implementation is hidden.

```
private void btn_SearchActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btn_SearchActionPerformed
        // TODO add your handling code here:
        String searchTerm = IM_tf_Search.getText().trim();
        JTable table = IM_tb_Inventory;
        if (searchTerm.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter a search term.", "Empty Search", JOptionPane.WARNING_MESSAGE);
            return;
        }
        List<?> searchResults = new ArrayList<>();
        searchResults = InventoryManagement.searchInventory(searchTerm);
        if (searchResults.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No results found for '" + searchTerm + "'.", "Search Results", JOptionPane.INFORMATION_MESSAGE);
            IM_tf_Search.setText("");
        } else {
            DefaultTableModel model = (DefaultTableModel) table.getModel();
            model.setRowCount(0);
            for (Object result : searchResults) {
                if (result instanceof Inventory) {
                    Inventory inv = (Inventory) result;
                    model.addRow(new Object[]{inv.getItemCode(),inv.getItemName(),inv.getQuantityInStock(),inv.getSupplierCode()});
                }
            }
        }
    }
```

Figure 2.2.2.1 Abstraction of Button and Field

- The user clicks on the button labelled as “Search” triggers `btn_SearchActionPerformed()`.
- The method calls `StaffAdminManagement.searchUser()`, where abstraction is applied by hiding the logic, thus user doesn’t see the internal implementation .

The `JDateChooser` component in Java GUI allows users to select a date without dealing with the complexity to formatting, validating or converting the date.

```
private com.toedter.calendar.JDateChooser startDate4;
private com.toedter.calendar.JDateChooser startDate5;
```

Figure 2.2.2.2 Declared of JDateChooser

```
startDate5 = new com.toedter.calendar.JDateChooser();
```

Figure 2.2.2.3 Initialize JDateChooser

```
private void applyReceiveDateFilter() {
        java.util.Date startDate = startDate4.getDate();
        java.util.Date endDate = endDate4.getDate();
        if (startDate == null || endDate == null) {
            JOptionPane.showMessageDialog(this, "Please select both start and end dates.", "Invalid Date Range", JOptionPane.WARNING_MESSAGE);
            btn_Distribute.setEnabled(true);
            btn_Distribute1.setEnabled(true);
            btn_Receive.setEnabled(true);
            btn_Receive1.setEnabled(true);
            return;
        }
        String startDateStr = new SimpleDateFormat("yyyy-MM-dd").format(startDate);
        String endDateStr = new SimpleDateFormat("yyyy-MM-dd").format(endDate);
        DefaultTableModel model = (DefaultTableModel) TracR_Rtb_DateFilter.getModel();
        model.setRowCount(0);
        List<String> filteredData = TrackingManagement.trackItemReceivedByDate(startDateStr, endDateStr);
        if (filteredData == null || filteredData.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No items received in this period.", "No Records Found", JOptionPane.INFORMATION_MESSAGE);
            btn_Distribute.setEnabled(true);
            btn_Distribute1.setEnabled(true);
            btn_Receive.setEnabled(true);
            btn_Receive1.setEnabled(true);
        } else {
            for (String data : filteredData) {
                String[] details = data.split(",");
                model.addRow(new Object[]{details[0], details[1], details[2], details[3]});
            }
        }
    }
```

Figure 2.2.2.4 Selected Date is Retrieved

- The user selects a date from the `JDateChooser` calendar component, triggering the `getDate()` method, which is used to retrieve the selected date in a structured format.
- `JDateChooser` manages the parsing, formatting and validation of dates without exposing its logic to the user.

2.2.3 Inheritance

Inheritance allows one class (subclass) to inherit attributes and methods from another class (superclass). It helps in reducing duplication of the code and helping in maintaining a hierarchical structure of classes.

```
public abstract class Entity {
    protected String id, name, email, contact, status;

    //method
}
```

Figure 2.2.3.1 Superclass – Entity

```
public class User extends Entity {
    private String password, userType;

    public User(String userID, String userName, String password, String userType, String userEmail, String userContact, String status) {
        super(userID, userName, userEmail, userContact, status);
        this.password = password;
        this.userType = userType;
    }
```

Figure 2.2.3.2 Subclass – User

- `User` class (subclass) inherits from `Entity` (superclass).
- Inherits fields such as `id`, `name`, `email`, `contact` and `status`, from the `Entity` class, and adds its own fields such as `password` and `userType`.

2.2.4 Polymorphism

Polymorphism in Java allows methods to have different behaviors based on the context. It can be achieved through method overriding or method overloading.

```
import java.util.Comparator;

public class InventoryComparator implements Comparator<Inventory> {
    @Override
    public int compare(Inventory i1, Inventory i2) {
        return i1.getItemCode().compareTo(i2.getItemCode());
    }
}
```

Figure 2.2.4 Method Override

- `InventoryComparator` class implements the `Comparator` interface.
- `compare()` method is polymorphic, it can be written differently based on the sorting method.
- This allows in flexible sorting based on different criteria such as item code, quantity or date.
- Different comparators can be created and each one behaves differently but using the same method (`compare`).

3.0 User Manual (GUI)

3.1 Login

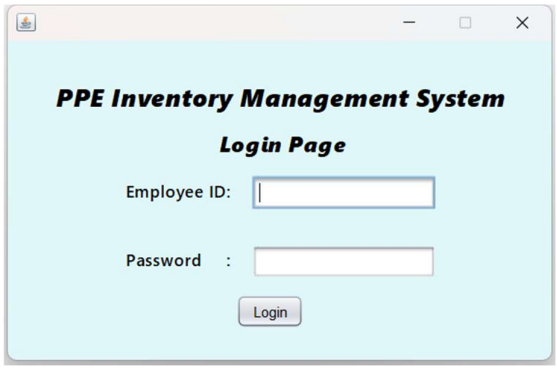

Figure 3.1.1 & Figure 3.1.2 Login Page

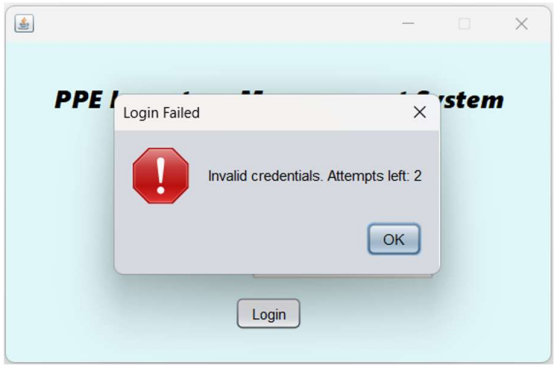

Figure 3.1.3 Login Fail Page

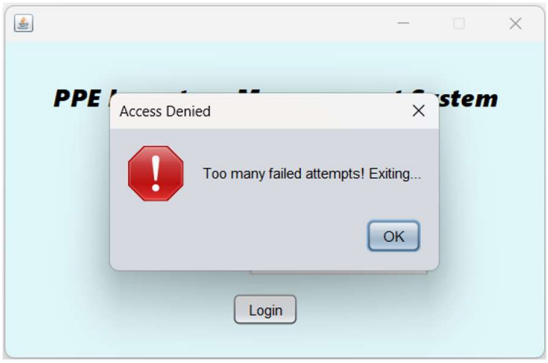

Figure 3.1.4 Exiting Login Page

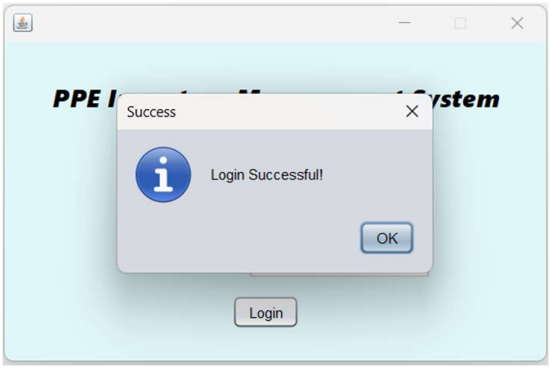

Figure 3.1.5 Login Success Page

3.2 Administrator

3.2.1 Administrator Dashboard

Figure 3.2.1.1 Administrator Dashboard

3.2.1 User and Entity Management (UEM)

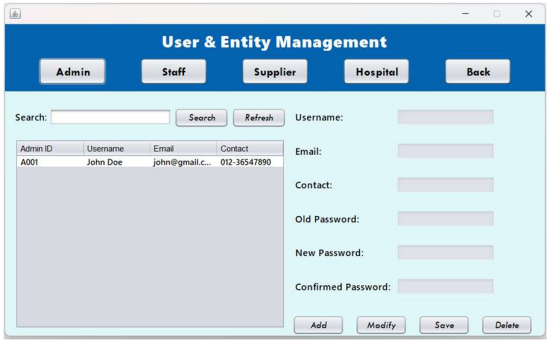

Figure 3.2.1.1 UEM Admin Page

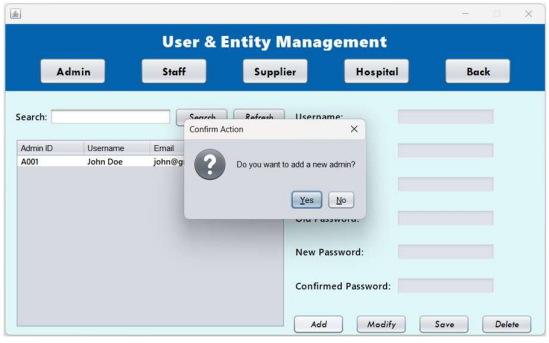

Figure 3.2.1.2 Add Admin Confirm Action

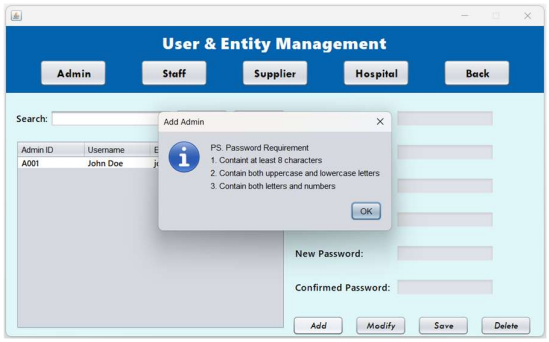

Figure 3.2.1.3 Password Setting Reminder

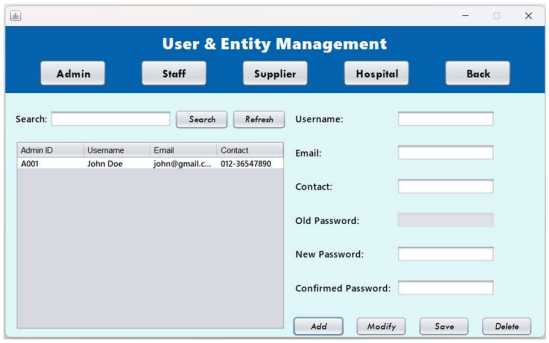

Figure 3.2.1.4 Insert New Admin Information

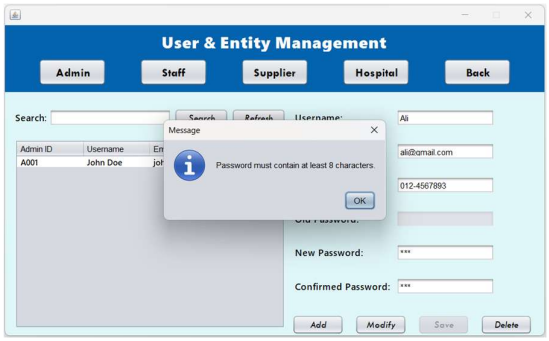

Figure 3.2.1.5 Password Error Reminder

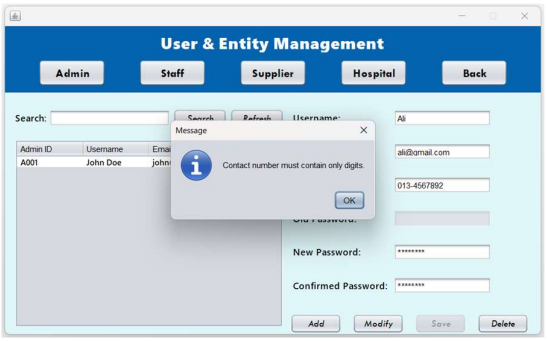

Figure 3.2.1.6 Contact Error Reminder

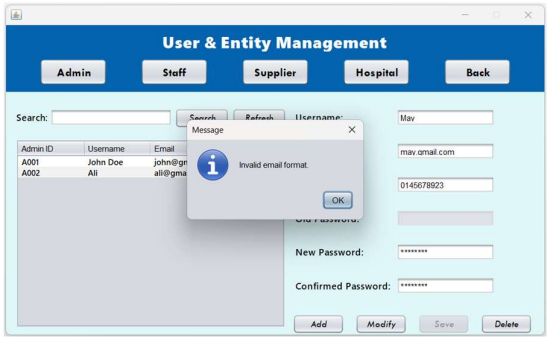

Figure 3.2.1.7 Email Error Reminder

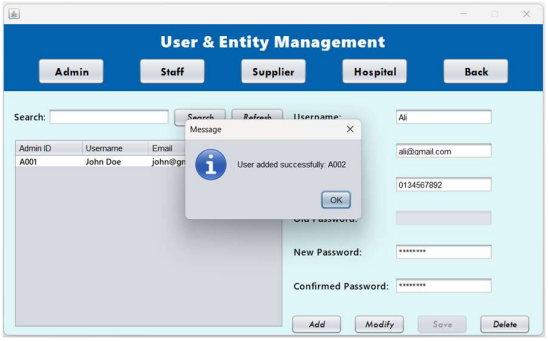

Figure 3.2.1.8 Add New Admin Successful Reminder

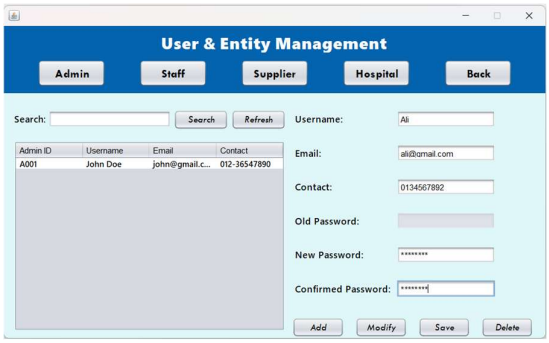

Figure 3.2.1.9 Insert New Admin Information Completed

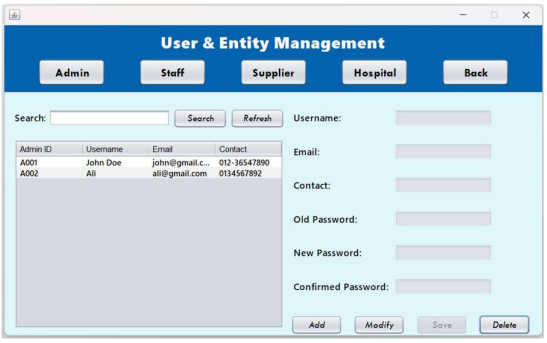

Figure 3.2.1.10 Updated Admin Table

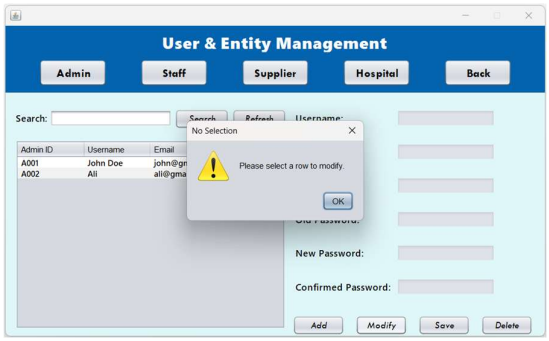

Figure 3.2.1.11 Modify Reminder

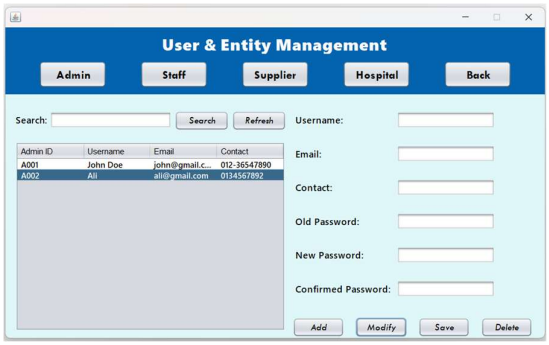

Figure 3.2.1.12 Select Admin for Modify

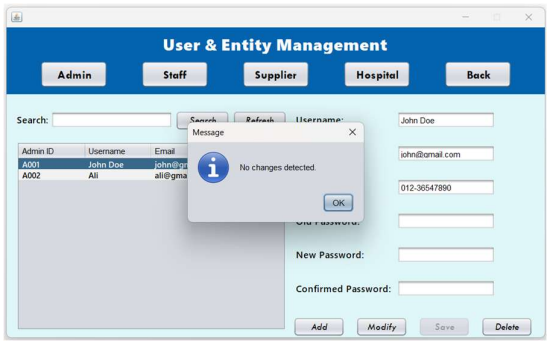

Figure 3.2.1.13 No Changes Detected Reminder

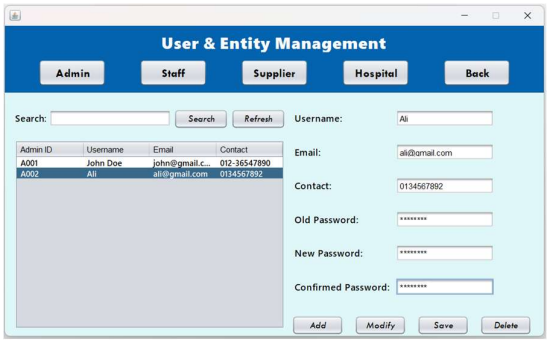

Figure 3.2.1.14 Modify Admin Information

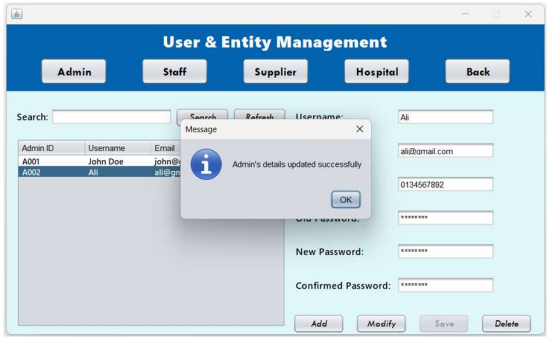

Figure 3.2.1.15 Modify Admin Information Completed

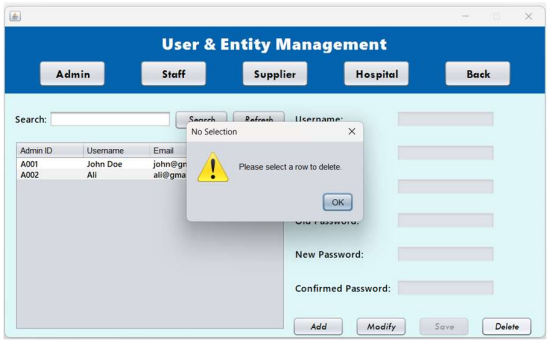

Figure 3.2.1.16 Select Admin for Delete

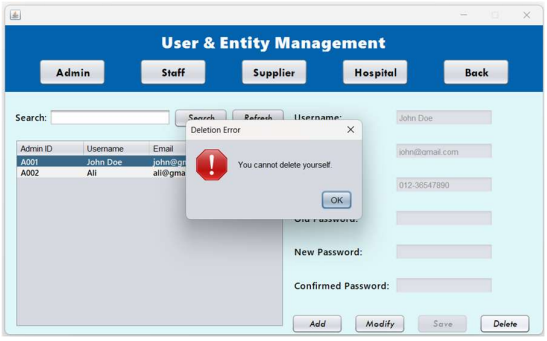

Figure 3.2.1.17 Error Message of Deleting Own Information

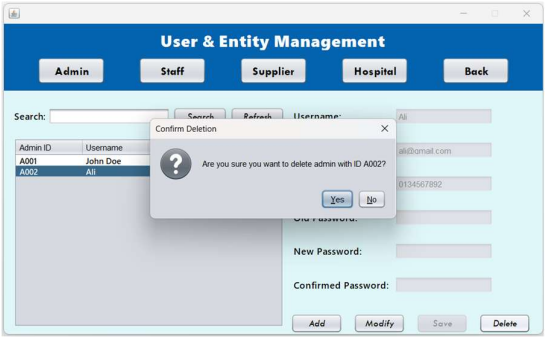

Figure 3.2.1.18 Confirmation Message for Delete

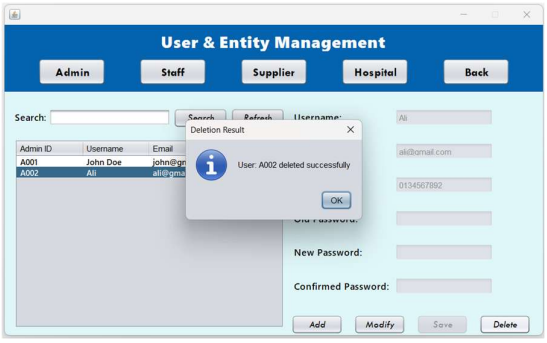

Figure 3.2.1.19 Delete Admin Completed

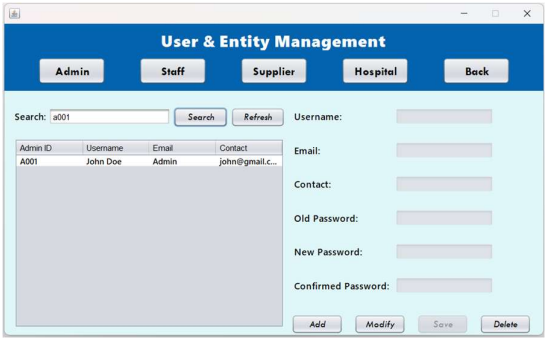

Figure 3.2.1.20 Search Admin

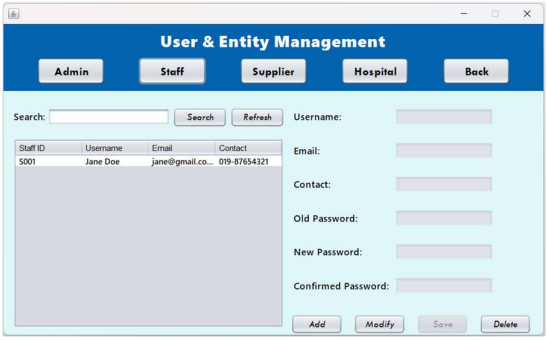

Figure 3.2.1.21 UEM Staff Page

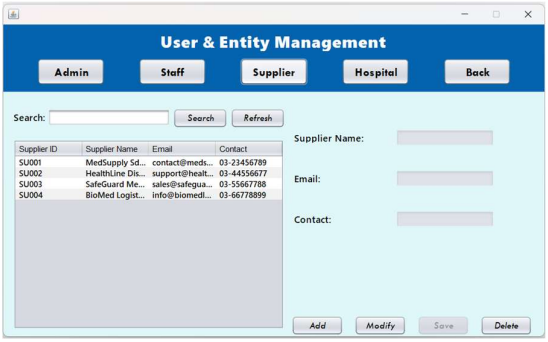

Figure 3.2.1.22 UEM Supplier Page

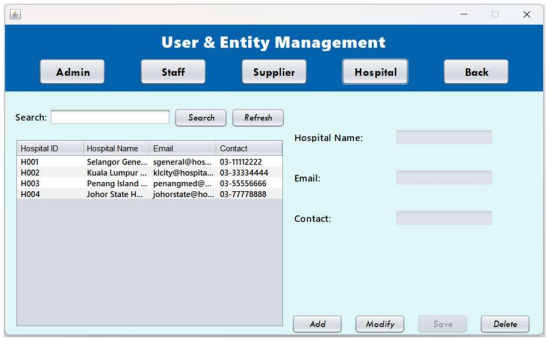

Figure 3.2.1.23 UEM Hospital Page

3.2.2 Inventory Management

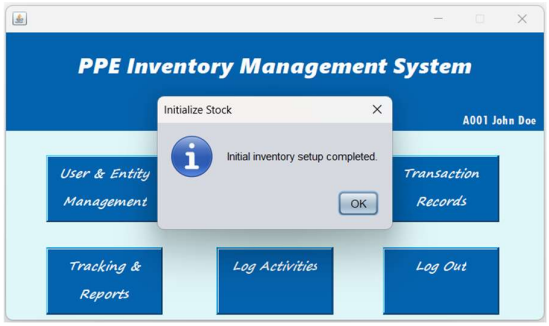

Figure 3.2.2.1 Initial Inventory Setup Reminder

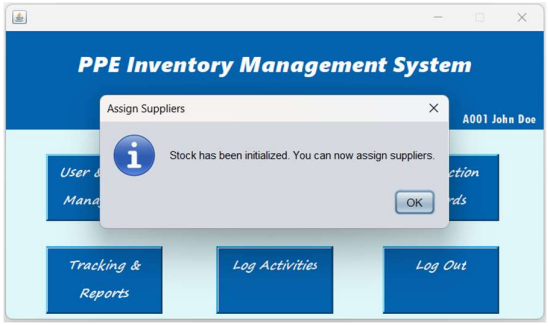

Figure 3.2.2.2 Assign Supplier Reminder

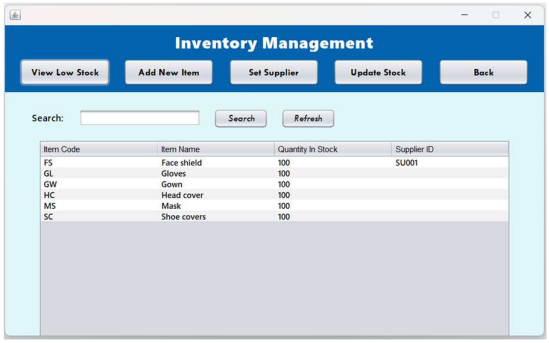

Figure 3.2.2.3 Inventory Management Page

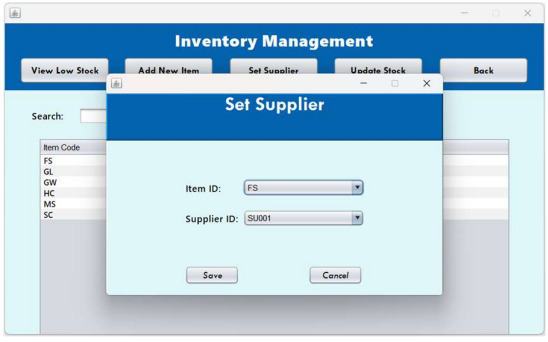

Figure 3.2.2.4 Assign Supplier Page

Figure 3.2.2.5 Assign Supplier Completed

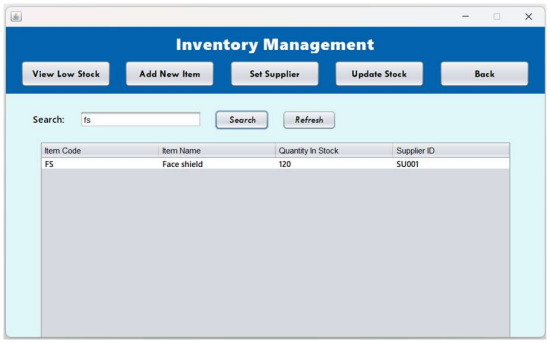

Figure 3.2.2.6 Search Item

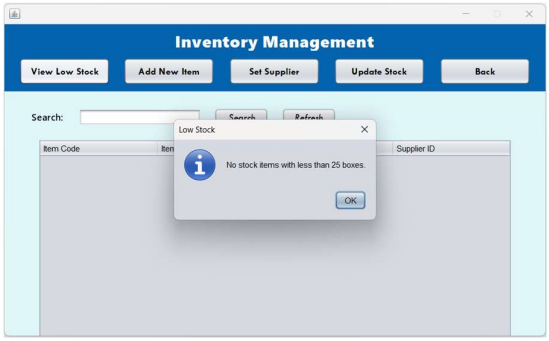

Figure 3.2.2.7 No Low Stock Reminder

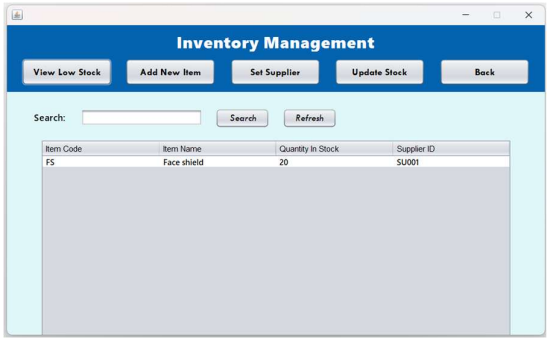

Figure 3.2.2.8 View Low Stock Page

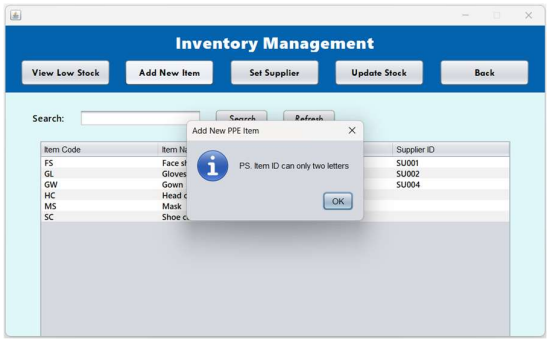

Figure 3.2.2.9 Add New Item Reminder

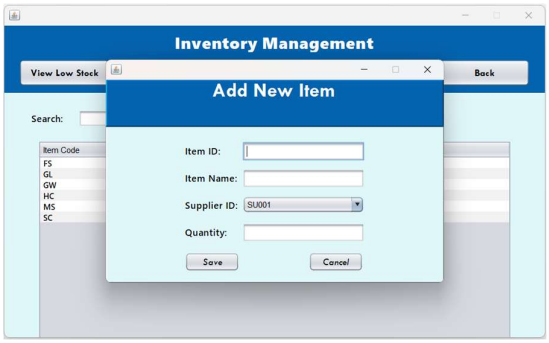

Figure 3.2.2.10 Add New Item Page

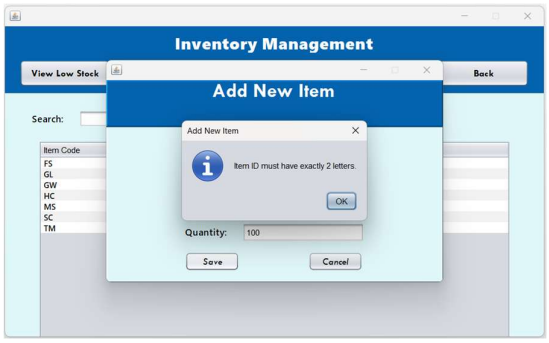

Figure 3.2.2.11 Error Message of Wrong Item Code Input Format

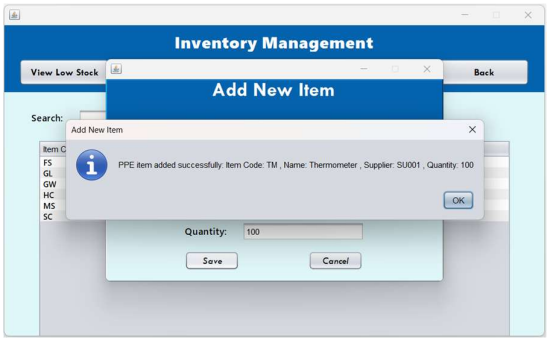

Figure 3.2.2.12 Add New Item Successfully Reminder

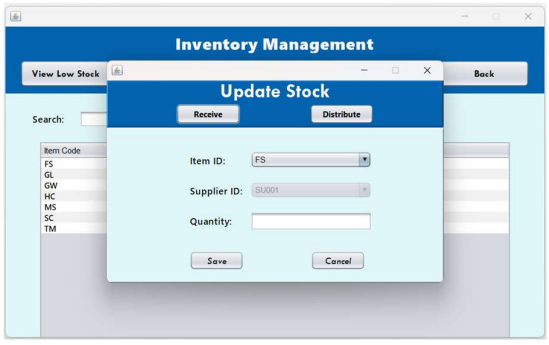

Figure 3.2.2.13 Update Stock Page of Receive

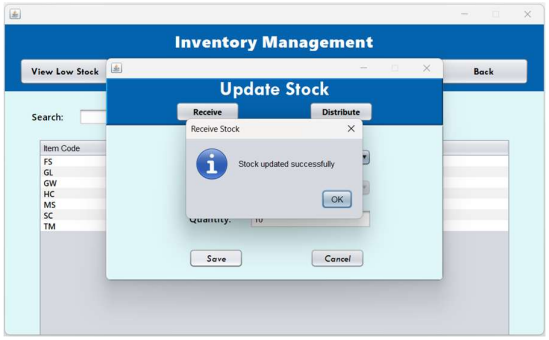

Figure 3.2.2.14 Update Stock Successfully of Receive Reminder

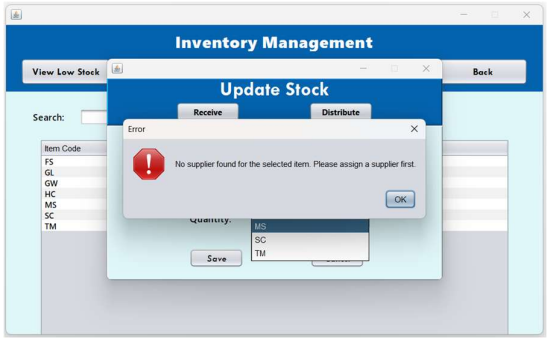

Figure 3.2.2.15 Error Message of None Supply Item

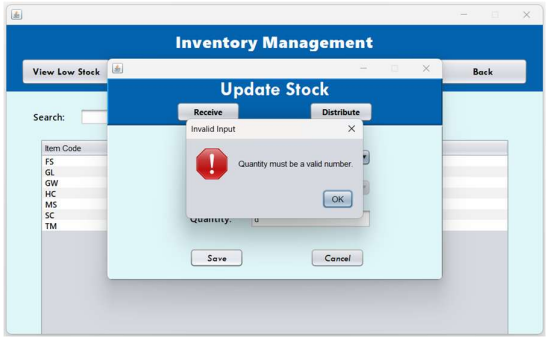

Figure 3.2.2.16 Error Message of Wrong Quantity Input Format

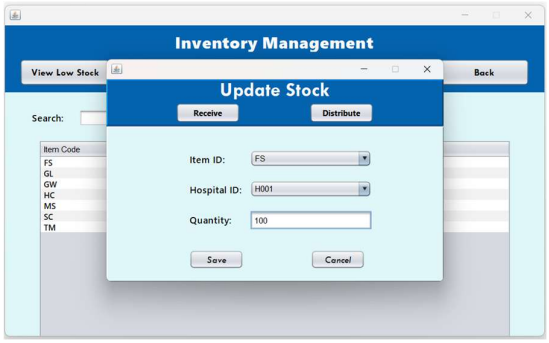

Figure 3.2.2.17 Update Stock Page of Distribute

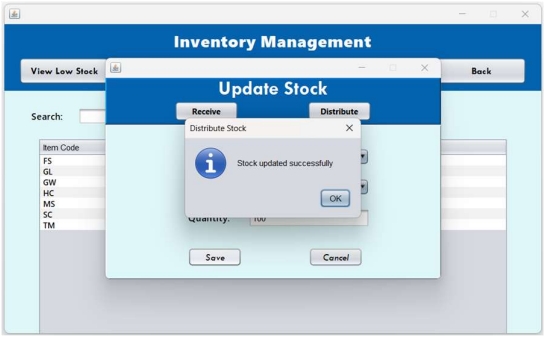

Figure 3.2.2.14 Update Stock Successfully of Distribute Reminder

3.2.3 Transaction Records

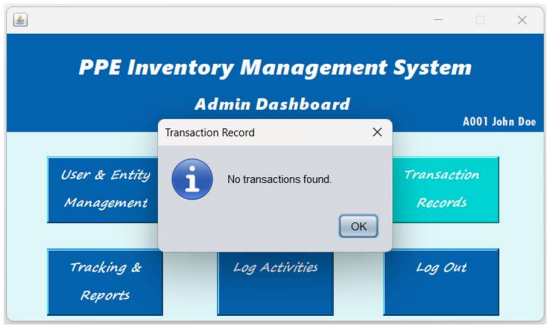

Figure 3.2.3.1 No Transaction Result Reminder

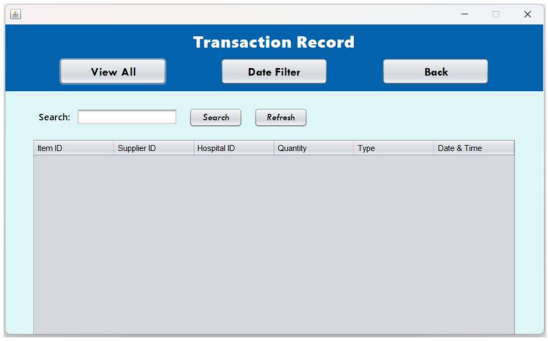

Figure 3.2.3.2 Empty Data Transaction Record Page

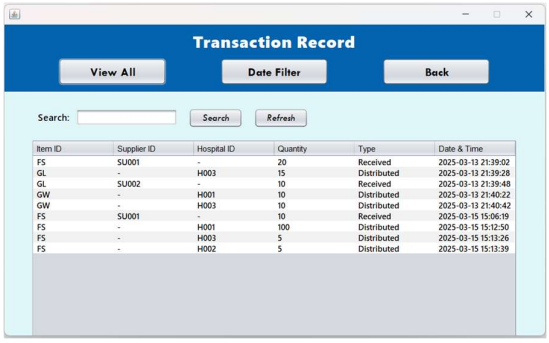

Figure 3.2.3.3 View All Page

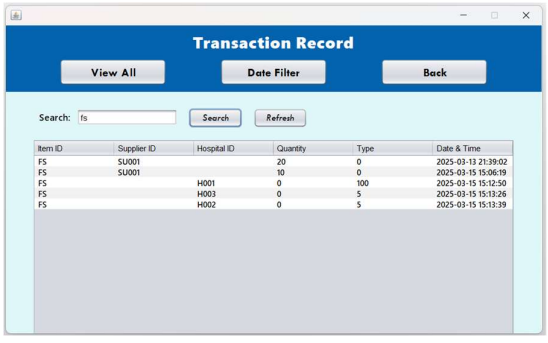

Figure 3.2.3.4 Search Transaction Record of Item

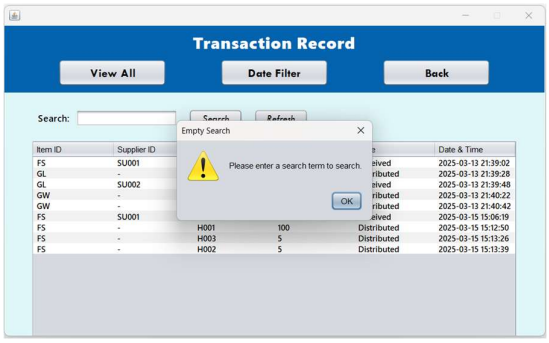

Figure 3.2.3.5 Error Message of Empty Search Input

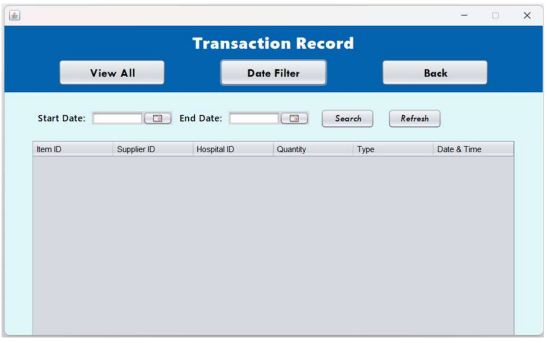

Figure 3.2.3.6 Date Filter Page

Figure 3.2.3.7 Input Date to View Record

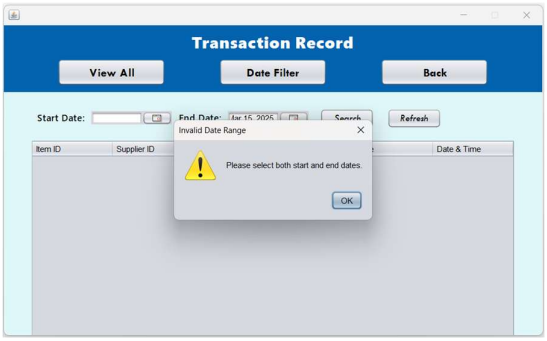

Figure 3.2.3.8 Error Message of Date Input Format

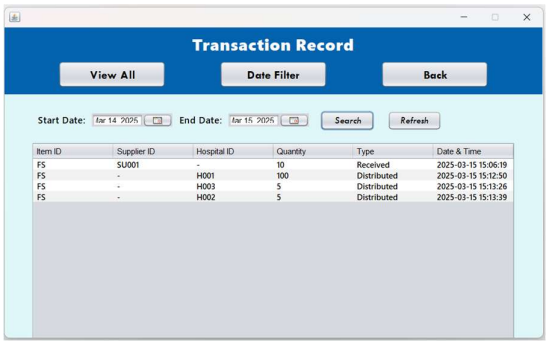

Figure 3.2.3.9 Date Filter Result

3.2.4 Tracking and Reports

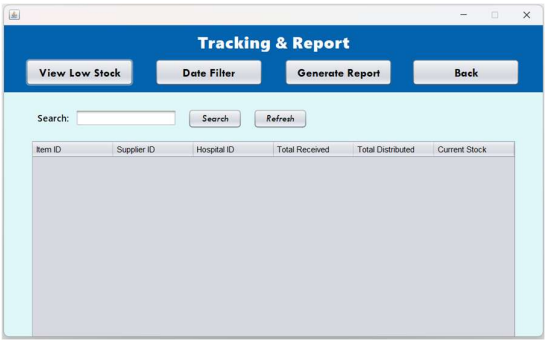

Figure 3.2.4.1 Empty Data Tracking & Report Homepage

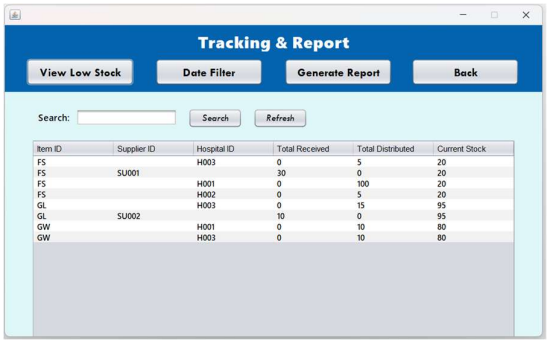

Figure 3.2.4.2 Tracking & Report Homepage

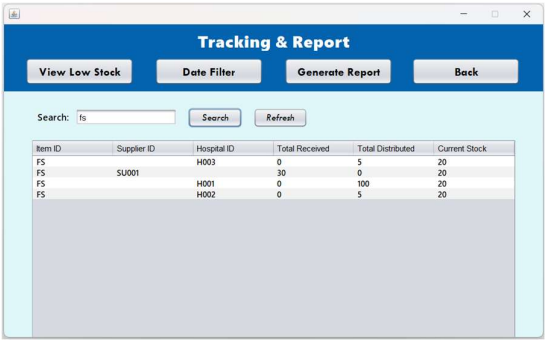

Figure 3.2.4.3 Search Tracking Record of Item

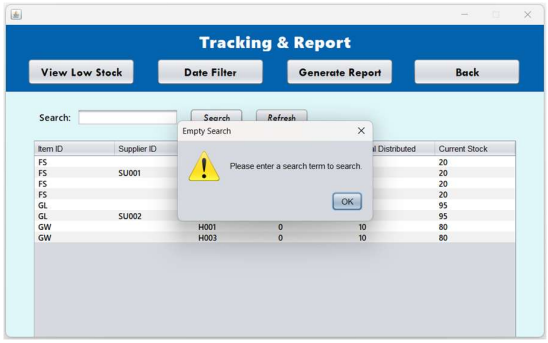

Figure 3.2.4.4 Error Message of Empty Search Input

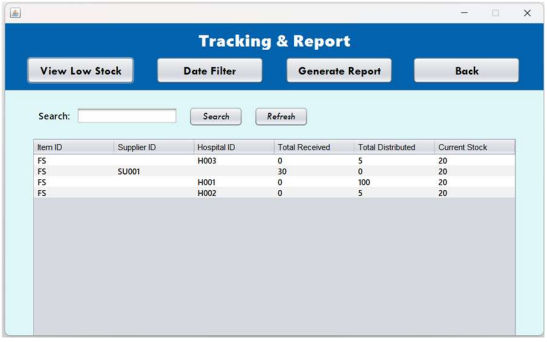

Figure 3.2.4.5 View Low Stock Page

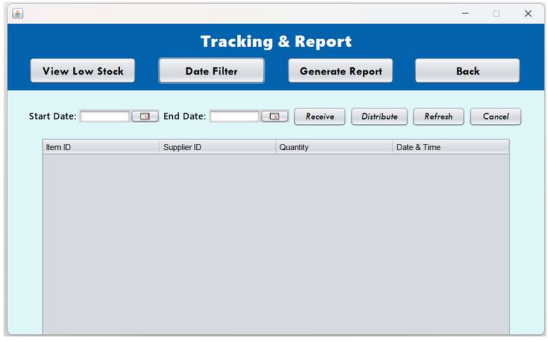

Figure 3.2.4.6 Date Filter Page

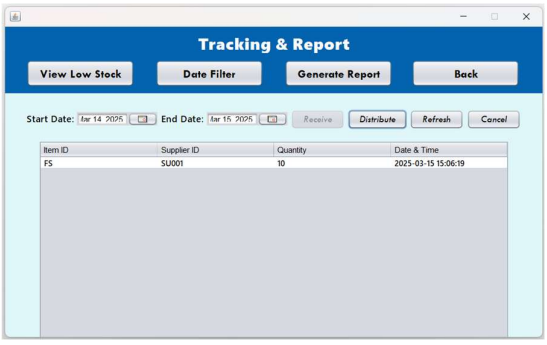

Figure 3.2.4.7 Date Filter Result of Receive

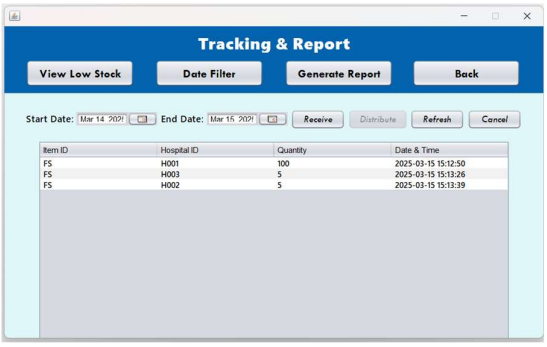

Figure 3.2.4.8 Date Filter Result of Distribute

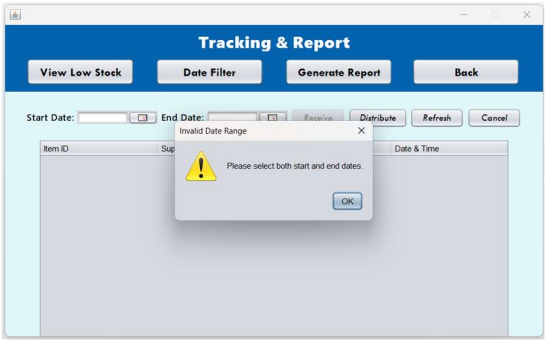

Figure 3.2.4.9 Error Message of Date Input Format

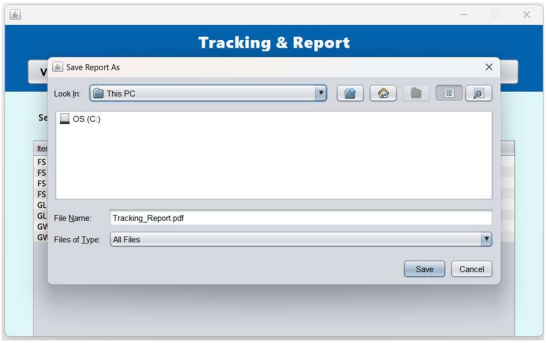

Figure 3.2.4.10 Generate Report Page

3.2.5 Log Activities

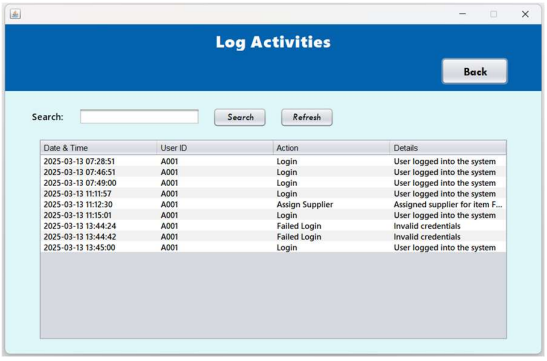

Figure 3.2.5.1 Log Activities Homepage

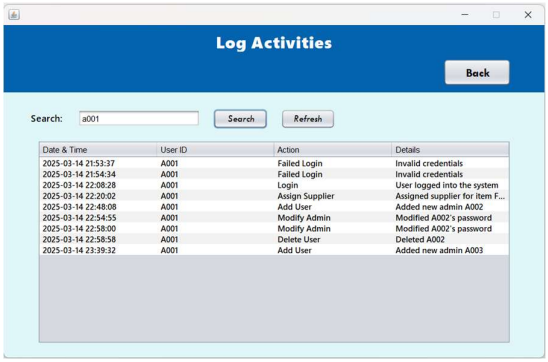

Figure 3.2.5.2 Search Log Activities of Users

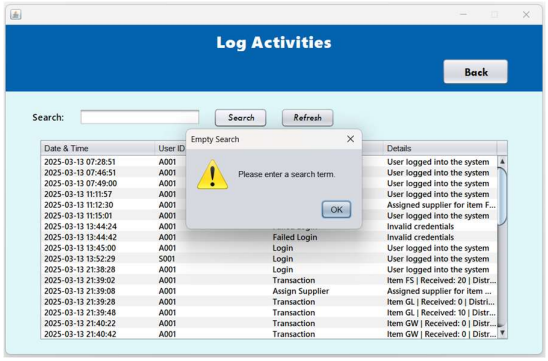

Figure 3.2.5.3 Error Message of Empty Search Input

3.3 Staff

3.3.1 Staff Dashboard

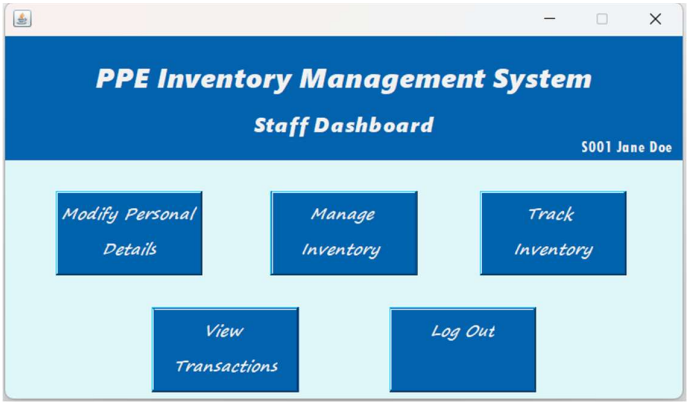

Figure 3.3.1 Staff Dashboard

3.3.2 Modify Personal Details

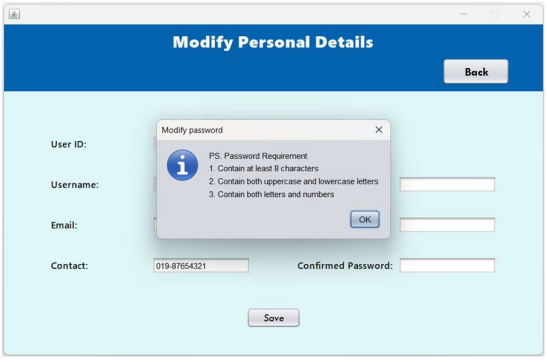

Figure 3.3.2.1 Password Setting Reminder

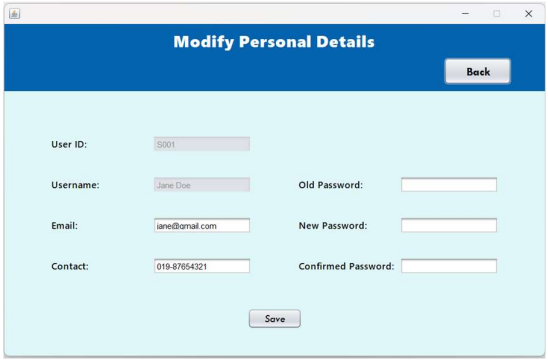

Figure 3.3.2.2 Modify Personal Details Page

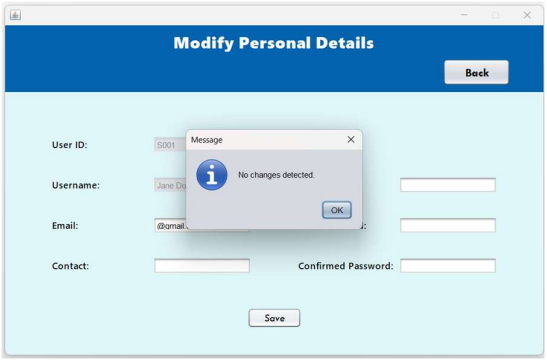

Figure 3.3.2.3 No Changes Detected Reminder

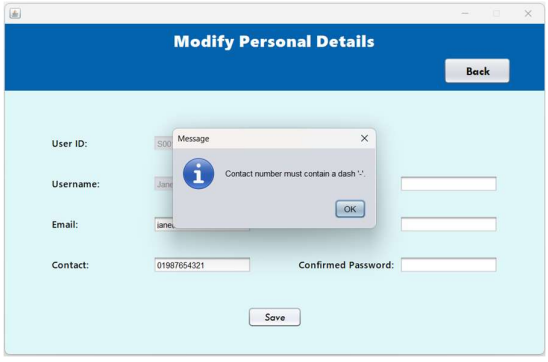

Figure 3.3.2.4 Contact Error Reminder

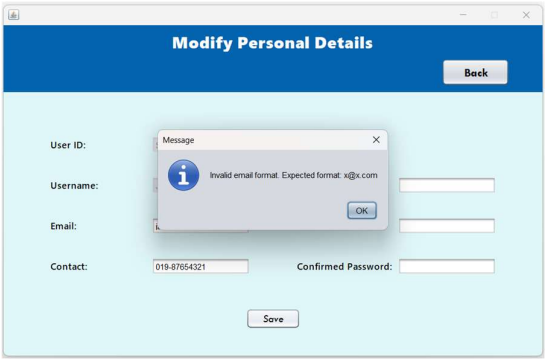

Figure 3.3.2.5 Email Error Reminder

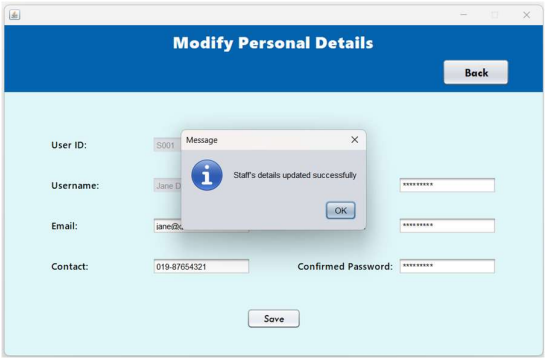

Figure 3.3.2.6 Modify Personal Details Successfully

3.3.3 Manage Inventory

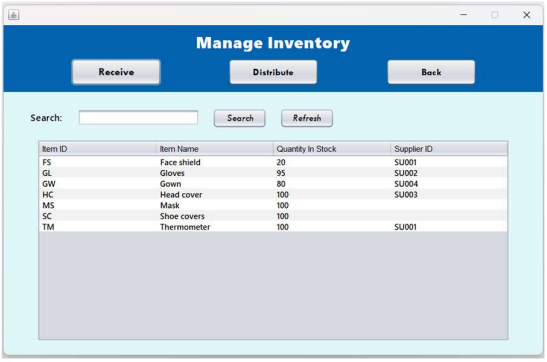

Figure 3.3.3.1 Manage Inventory Homepage

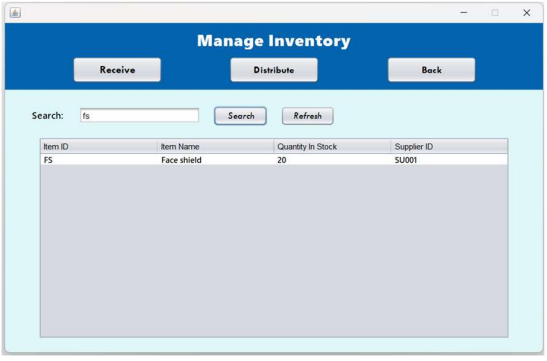

Figure 3.3.3.2 Search Item Stock

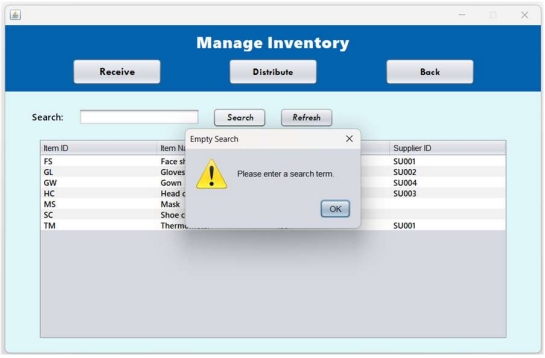

Figure 3.3.3.3 Error Message of Empty Search Input

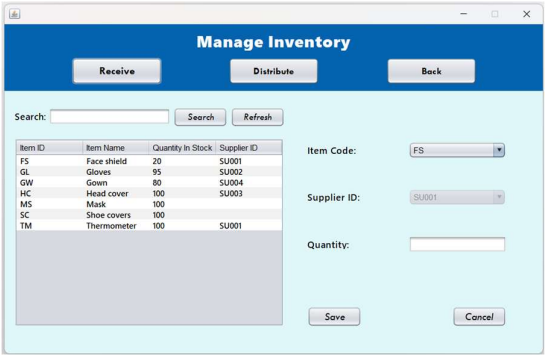

Figure 3.3.3.4 Receive Page

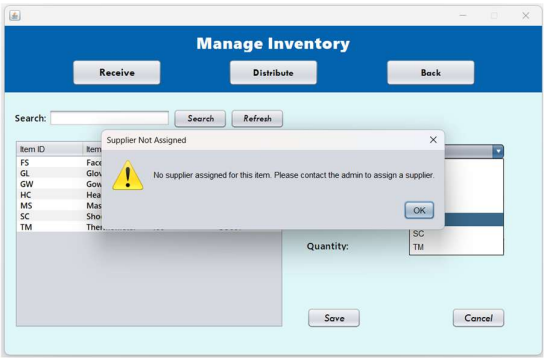

Figure 3.3.3.5 Error Message of None Supply Item

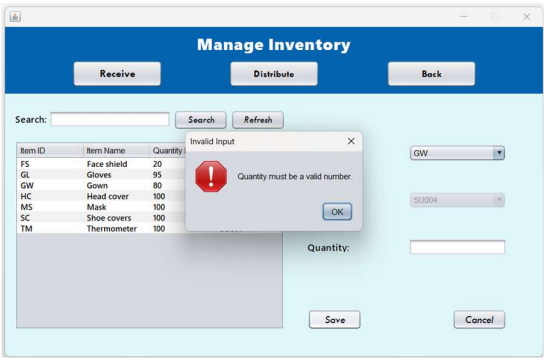

Figure 3.3.3.6 Error Message of Wrong Quantity Input Format

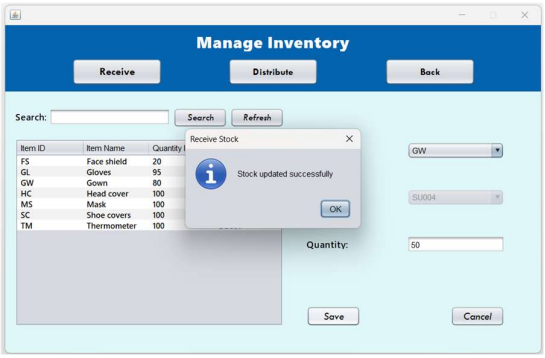

Figure 3.3.3.7 Update Stock Successfully of Receive Reminder

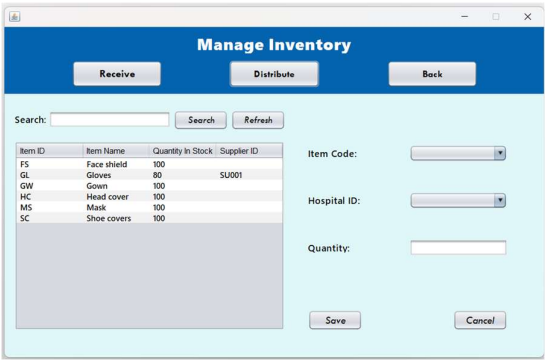

Figure 3.3.3.8 Distribute Page

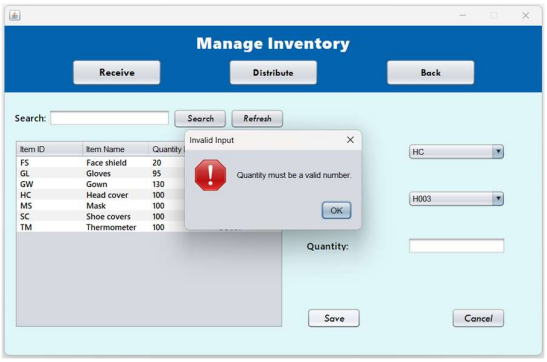

Figure 3.3.3.9 Error Message of Wrong Quantity Input Format

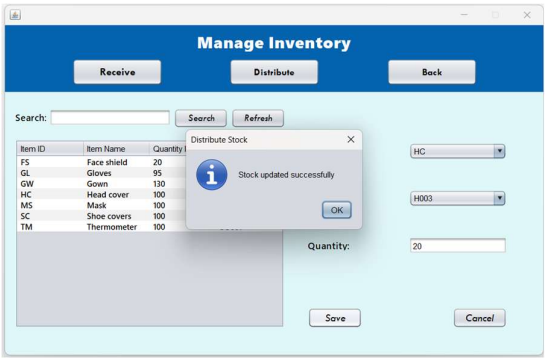

Figure 3.3.3.10 Update Stock Successfully of Distribute Reminder

3.3.4 Track Inventory

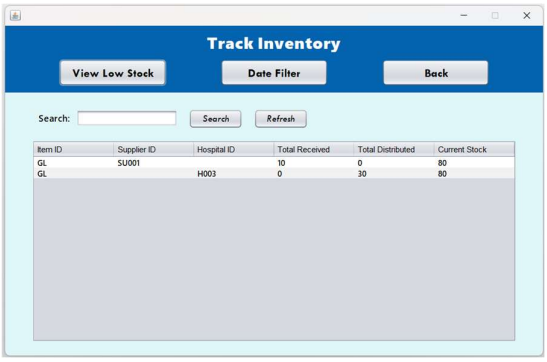

Figure 3.3.4.1 Track Inventory Page

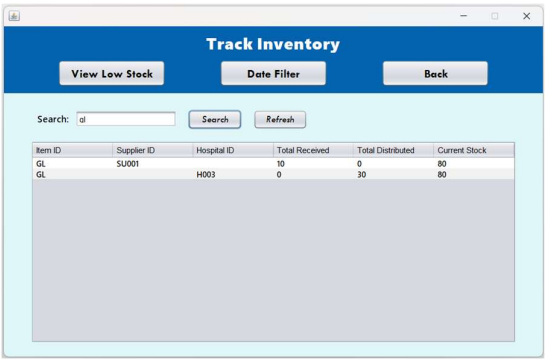

Figure 3.3.4.2 Search Tracking Inventory Record

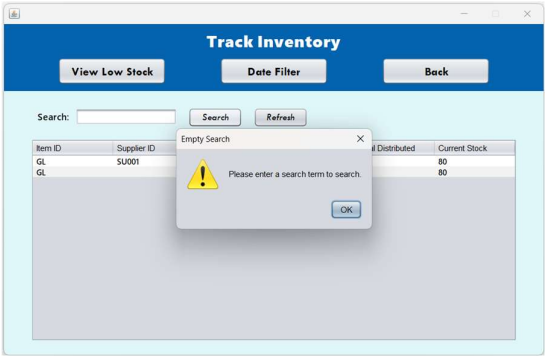

Figure 3.3.4.3 Error Message of Empty Search Input

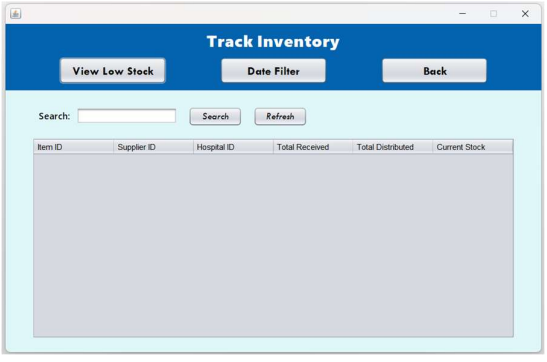

Figure 3.3.4.4 View Low Stock Page

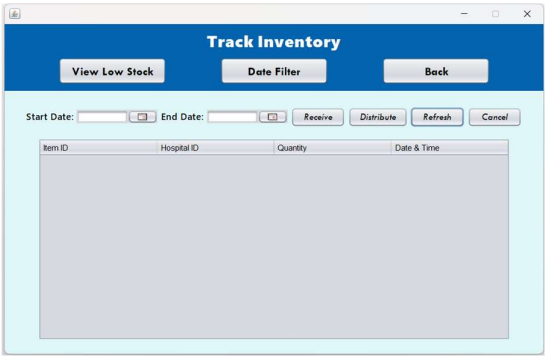

Figure 3.3.4.5 Date Filter Page

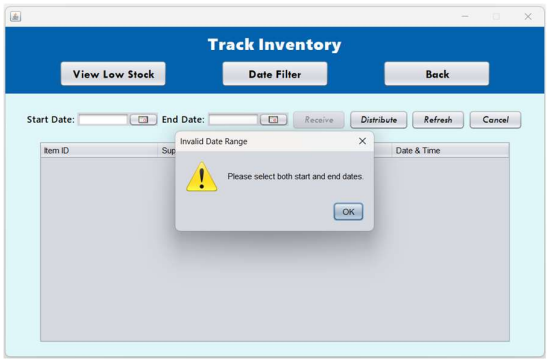

Figure 3.3.4.6 Error Message of Date Input Format

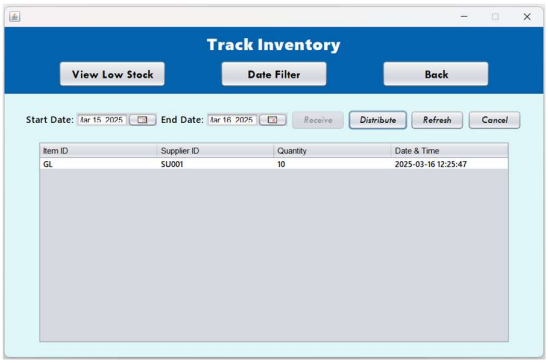

Figure 3.3.4.7 Date Filter Result of Receive

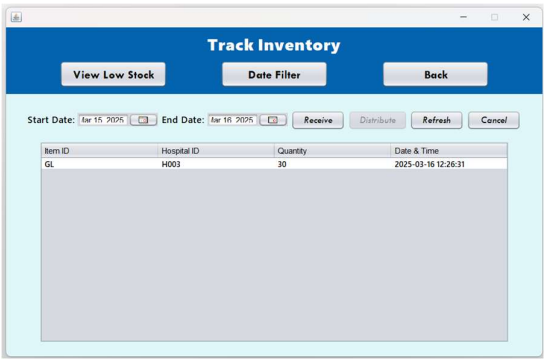

Figure 3.3.4.8 Date Filter Result of Distribute

3.3.5 View Transactions

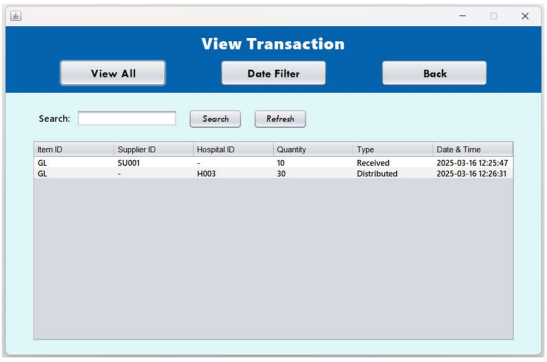

Figure 3.3.5.1 View Transaction Page

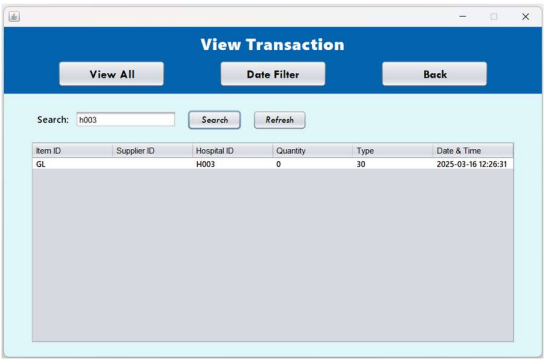

Figure 3.3.5.2 Search Transaction Record

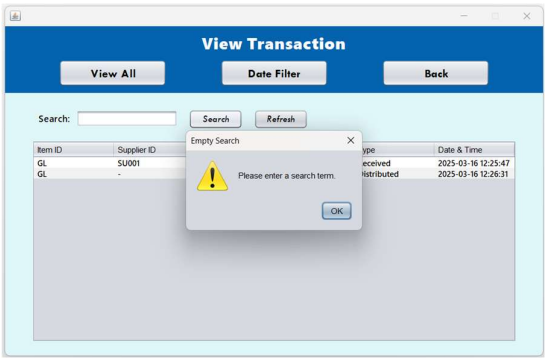

Figure 3.3.5.3 Error Message of Empty Search Input

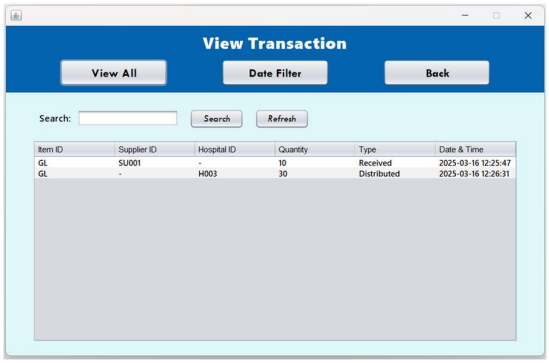

Figure 3.3.5.4 View All Page

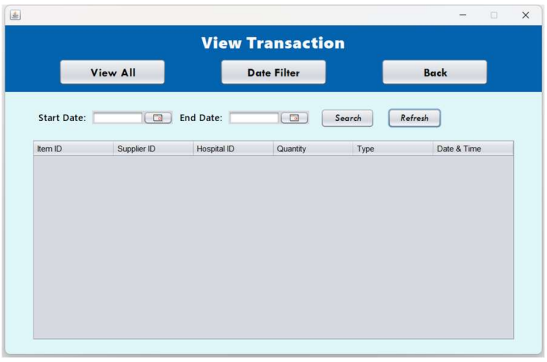

Figure 3.3.5.5 Date Filter Page

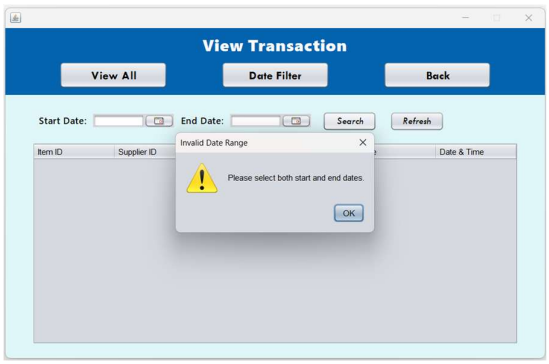

Figure 3.3.5.6 Error Message of Date Input Format

Figure 3.3.5.7 Date Filter Results

3.4 Logout

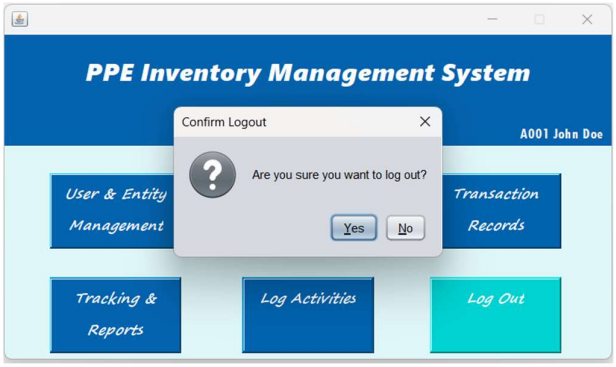

Figure 3.4 Logout Page

4.0 Conclusion

The development of Personal Protective Equipment (PPE) Inventory Management System has applied the concepts and principles of Object-Oriented Programming (OOP). This system provides main functionalities such as user management, inventory management, transaction tracking, and generating reports, thus supporting an efficient management of PPE items.

Throughout the implementation, OOP concepts such as class and object, and OOP principles such as encapsulation, inheritance, polymorphism and abstraction were applied to the system, ensuring that the system is modular, maintainable and reusable. The system uses a user-friendly GUI interface, enabling both administrator and staff interact with the system easily.

By developing this system, the understanding of Java Programming and GUI development is enhanced.

5.0 References

Abba, I. (2022, February 7). *For Loop in Java + ForEach Loop Syntax example*. freeCodeCamp.org. https://www.freecodecamp.org/news/for-loop-in-java-foreach-loop-syntax-example/

baeldung. (2018, August 13). *Read a File into an ArrayList | Baeldung*. Www.baeldung.com. https://www.baeldung.com/java-file-to-arraylist

baeldung. (2018, October 11). *Baeldung*. Baeldung. https://www.baeldung.com/java-datetimeformatter

Format an Integer using Java String Format. (n.d.). Stack Overflow. https://stackoverflow.com/questions/6034523/format-an-integer-using-java-string-format

GeeksforGeeks. (2020, November 19). *LocalDate isBefore() method in Java with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/localdate-isbefore-method-in-java-with-examples/

GeeksforGeeks. (2022, February 21). *Read file into an array in java*. GeeksforGeeks. https://www.geeksforgeeks.org/read-file-into-an-array-in-java/

GeeksforGeeks. (2023, April 23). *LocalDate parse() method in Java with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/localdate-parse-method-in-java-with-examples/

GeeksforGeeks. (2024, February 7). *Java program to capitalize the first letter of each word in a string*. GeeksforGeeks. https://www.geeksforgeeks.org/java-program-to-capitalize-the-first-letter-of-each-word-in-a-string/

GeeksforGeeks. (2025, January 4). *Different ways of Reading a text file in Java*. GeeksforGeeks. https://www.geeksforgeeks.org/different-ways-reading-text-file-java/

How to add leading zeros to integers in Java ? String Left padding example program. (n.d.). https://javarevisited.blogspot.com/2013/02/add-leading-zeros-to-integers-Java-String-left-padding-example-program.html#goog_rewarded

How to get System time in Netbeans? (n.d.). Stack Overflow. https://stackoverflow.com/questions/7962200/how-to-get-system-time-in-netbeans

Intact Abode. (2020, January 31). *How to add DatePicker, Calender in palette in Netbeans - Java Swing - Intact Abode* [Video]. YouTube. https://www.youtube.com/watch?v=osyM_NugYgc

Java | ArrayList | .set() | Codecademy. (2023, January 19). Codecademy. https://www.codecademy.com/resources/docs/java/array-list/set

Java Tech Solutions. (2024, February 2). *How to create PDF file using iText Library using Spring Boot | Java 17 | JavaTechSolutions* [Video]. YouTube. https://www.youtube.com/watch?v=_FUCwZ8Hf8w

Knowledge to Share. (2021, January 14). *How to make a JTable not editable in JAVA Swing using NetBeans* [Video]. YouTube. https://www.youtube.com/watch?v=bcmg4WvnOVg

Max O'Didily. (2019, April 16). *How to Delete a Record from a File with Java* [Video]. YouTube. https://www.youtube.com/watch?v=NceIYifVAQQ

Pankaj. (2022, August 4). *Java FileWriter Example*. DigitalOcean. https://www.digitalocean.com/community/tutorials/java-filewriter-example

Singh, C., & Singh, C. (2024, June 1). How to sort ArrayList in Java. *BeginnersBook* -. https://beginnersbook.com/2013/12/how-to-sort-arraylist-in-java/

Sort ArrayList of objects by specific element. (n.d.). Stack Overflow. https://stackoverflow.com/questions/42121624/sort-arraylist-of-objects-by-specific-element