

**De La Salle University • College of Computer Studies**

**SystemScape**

**UI Standards, Development Standards, ER Diagram, and Class Diagram**

Name (last name first) : Angeles, Antonio Mariano

Coquilla, Philip Bryan

Cote, Christian Gabriel

Fernandez, Ryan Austin

Poblete, Clarisse Felicia

Quindoza, Rissa Marie Grace

Salceda, Juan Francesco

Tan, Shayane

Uy, Mervin Stewart

Velez, Gio Anton

Section : S19A

Date of Submission : February 16, 2015

**Table of Contents**

A. UI Standards / A-1

A.1. Purpose of the Document / A-1

A.2. SystemScape UI Design Standards / A-1

1. Color scheme / A-1
2. UI Font Family / A-1
3. UI Elements Naming Convention / A-1
4. Layout of UI / A-2
5. Messages / A-6
6. Titles and Captions / A-7
7. Command Buttons / A-7
8. Sort order for Lists / A-7
9. Usability Principles / A-7

B. Development Standards / B-1

B.1. Programming language, compiler, editor, debugger / B-1

B.2. Logic and UI Development Sequence / B-1

B.3. Top-Down or Bottom-Up Development / B-1

B.4. Coding Standards Compliance / B-1

B.5. Error Handling and Security / B-1

B.6. Programming Strategy / B-1

B.7. Version Control / B-1

B.8. Revisions and Revision Control / B-2

B.9. Non-standard Language Feature Use / B-2

B.10. Test Cases / B-2

B.11. Debugging and Code Check-In / B-2

B.12. Integration Testing / B-2

B.13. Code Review and Inspection / B-2

C. ER Diagram / C-1

D. Class Diagram / D-1

**A. User Interface Standards**

**A.1. Purpose of the Document**

This document defines the user interface design standards that the team, SystemScape has settled. The user interface designs exposed in this document shall be applied carefully in the development of the software for CAI-STA to ensure consistency and comprehensibility.

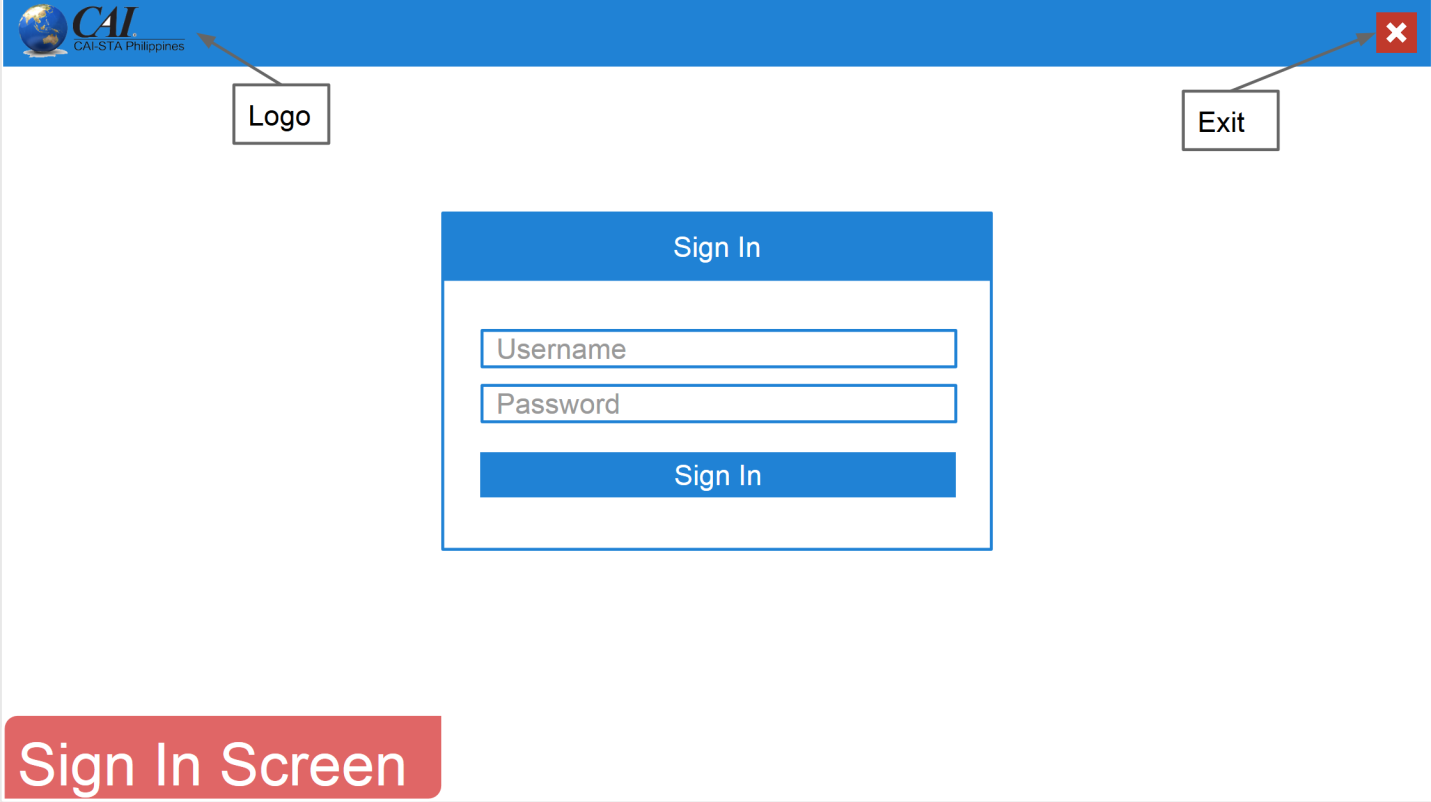
**A.2.** **SystemScape UI Design Standards**

1. Color scheme

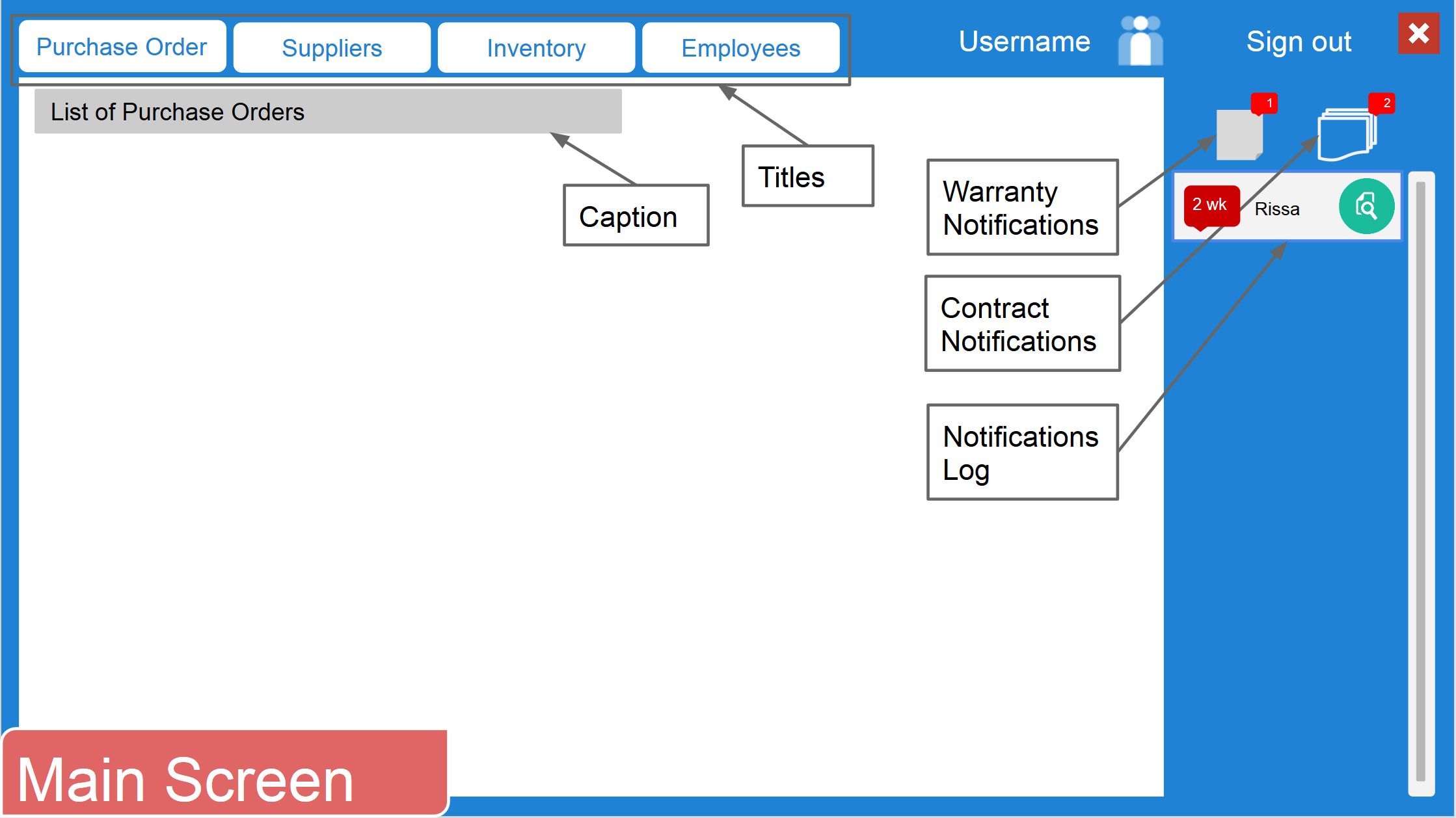


**Figure A-1. – UI Color Scheme**

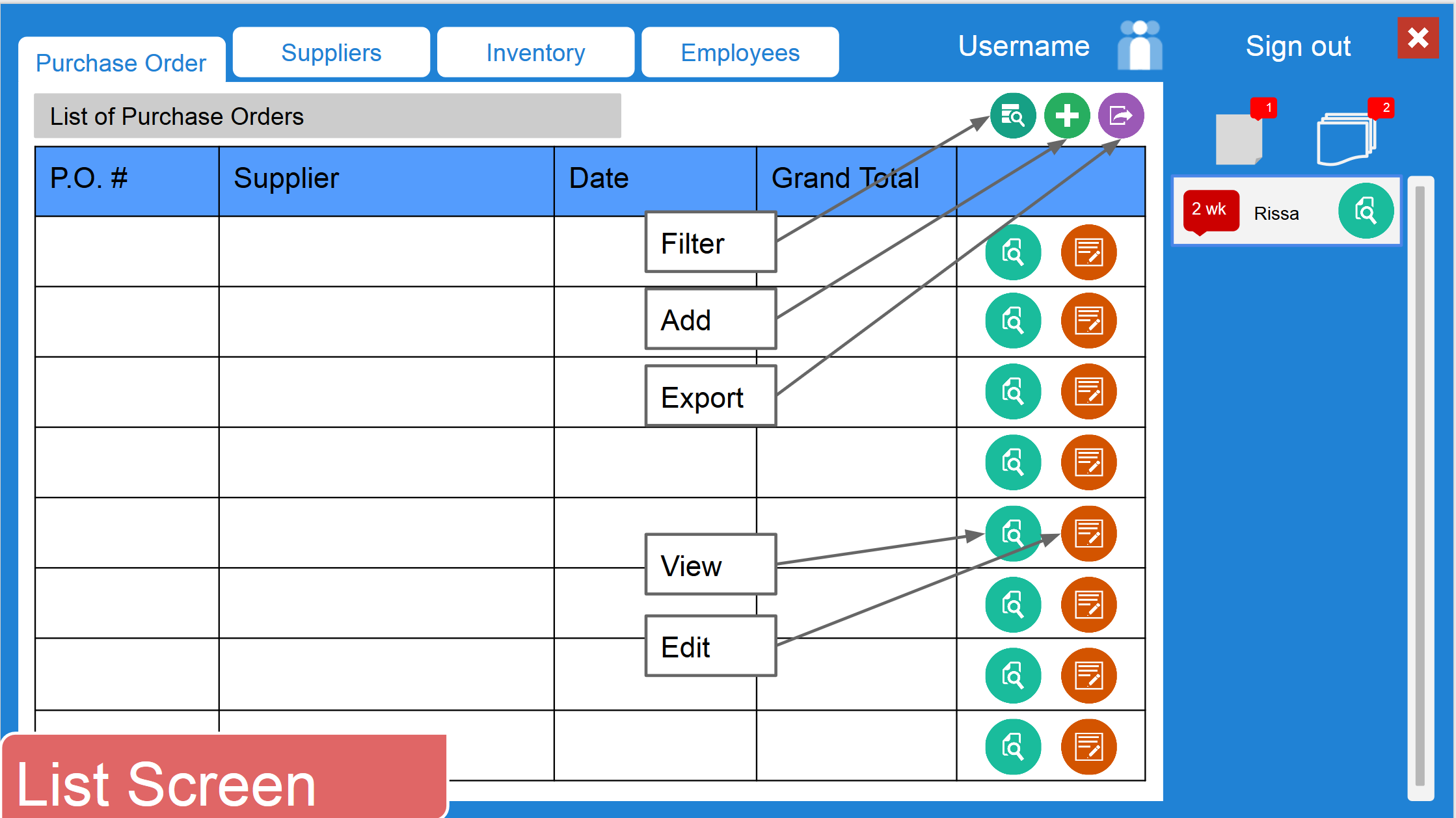
1. UI Font Family to be used will be Arial.
2. UI Elements Naming Convention
   1. All names will be in English.
   2. Button names will be written following the standard English format.
   3. Button text will start with an uppercase letter followed by lowercase letters.
   4. The word “Submit” will be used during the process of submission to indicate the action to save the information into the database.
   5. The word “Ok” will be used to indicate that the user has understood the message.
   6. The iconDelete.pngwill be used to close the program completely.
   7. The icon will be used to close the pop-up windows for viewing, adding, and editing of information.
   8. The word “Yes” will be used to indicate that user is giving the system permission to perform an event.
   9. The word “Cancel” will be used to indicate that the event will not be performed.
   10. The word “Logout” will be used to indicate that the access of a user for a specific part of the program is no longer needed.
   11. The word “Delete” will be used to remove the identified data and associated data from view, but retain its records in the database.
3. Layout of UI



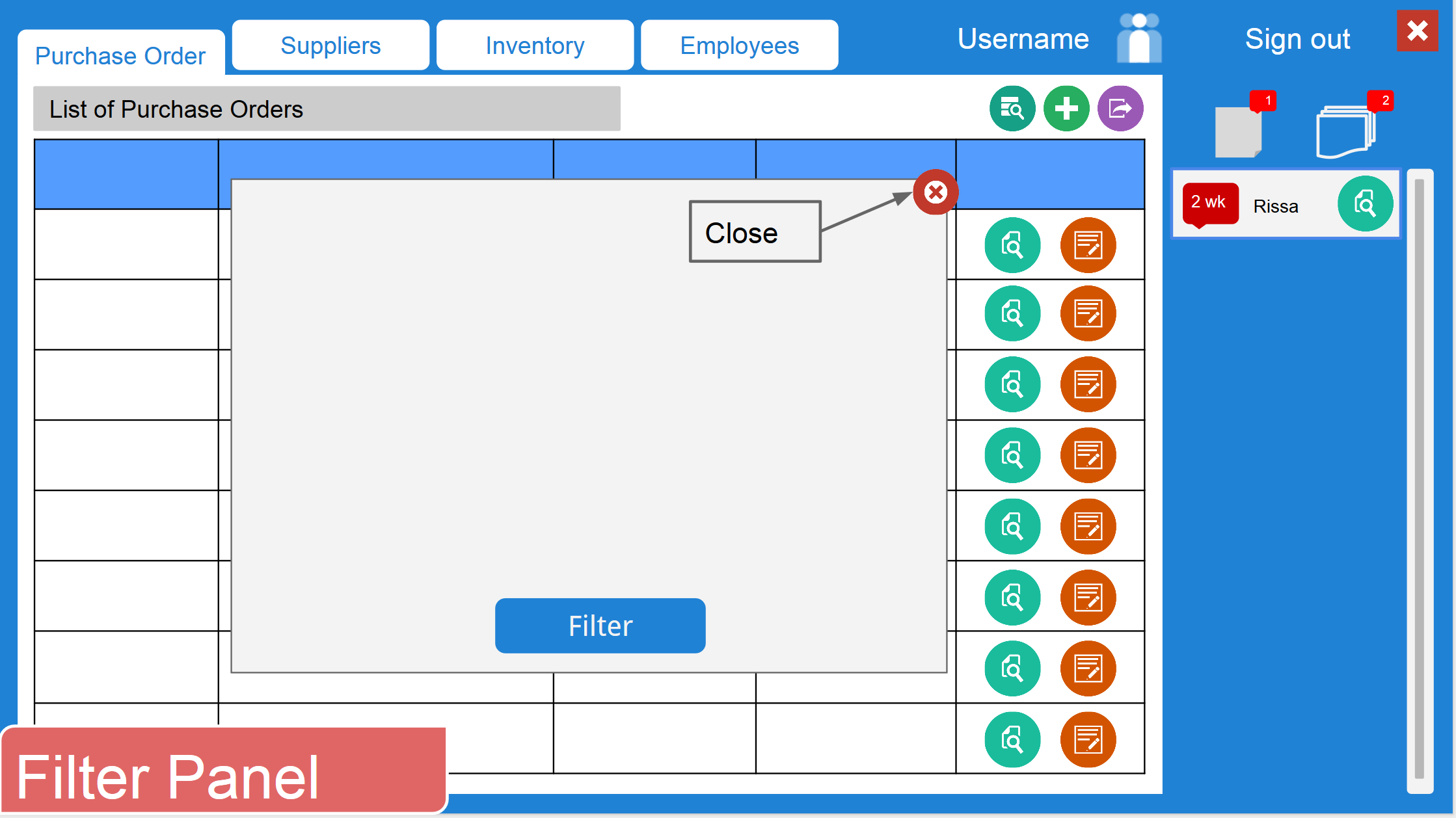
**Figure A-2. - Sign In Screen**



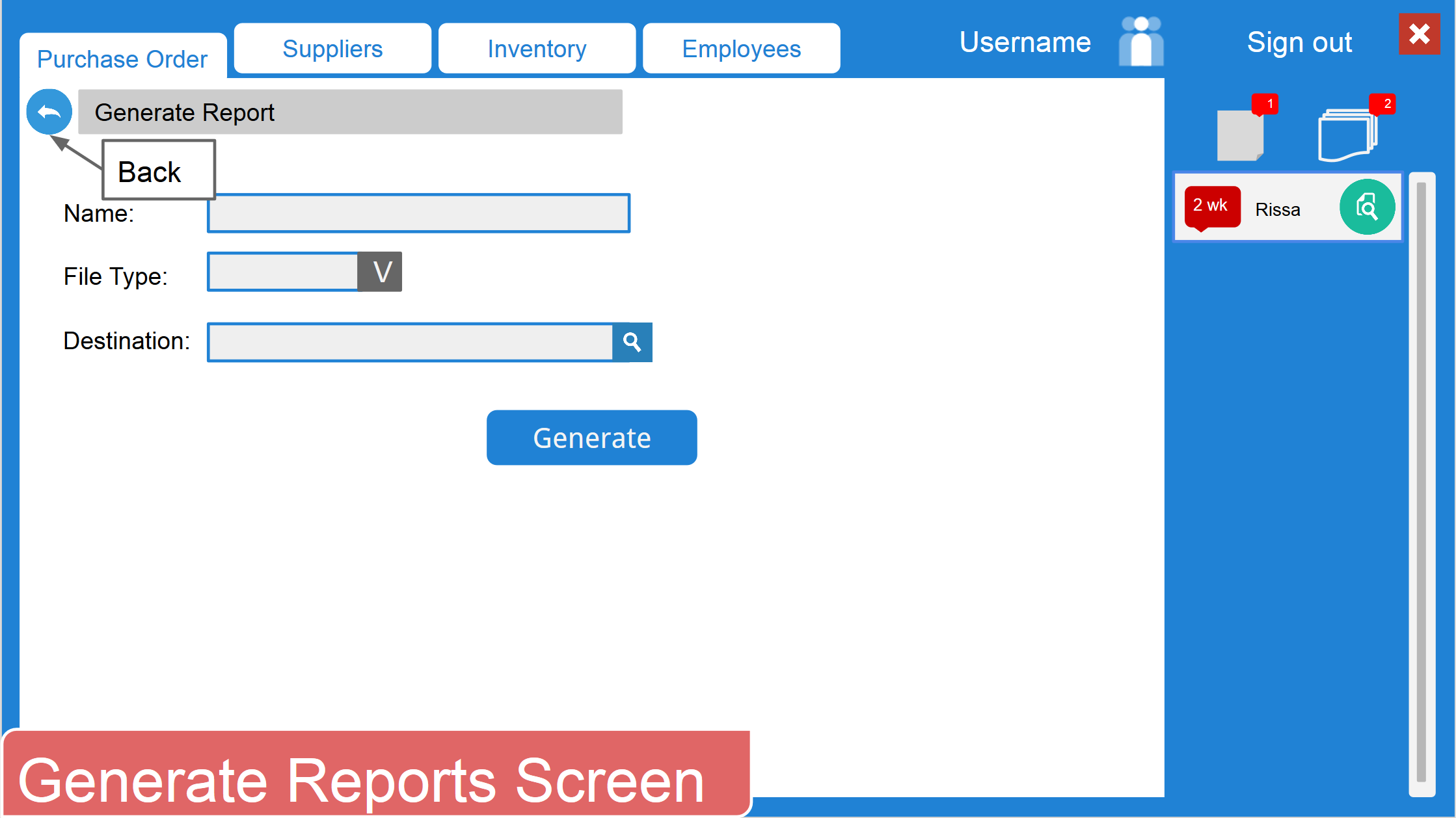
**Figure A-3. - Main Screen**



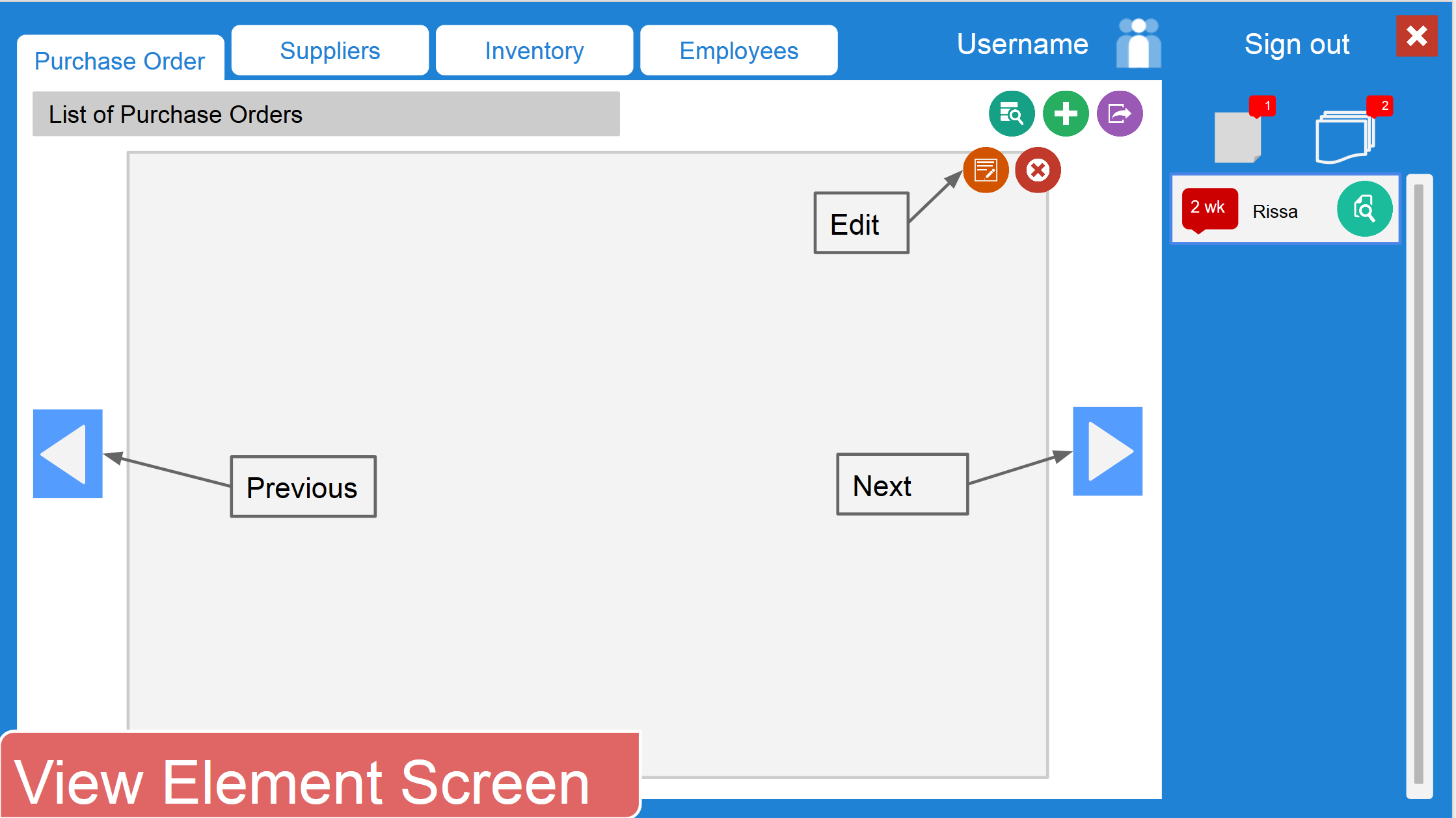
**Figure A-4. - List Screen**



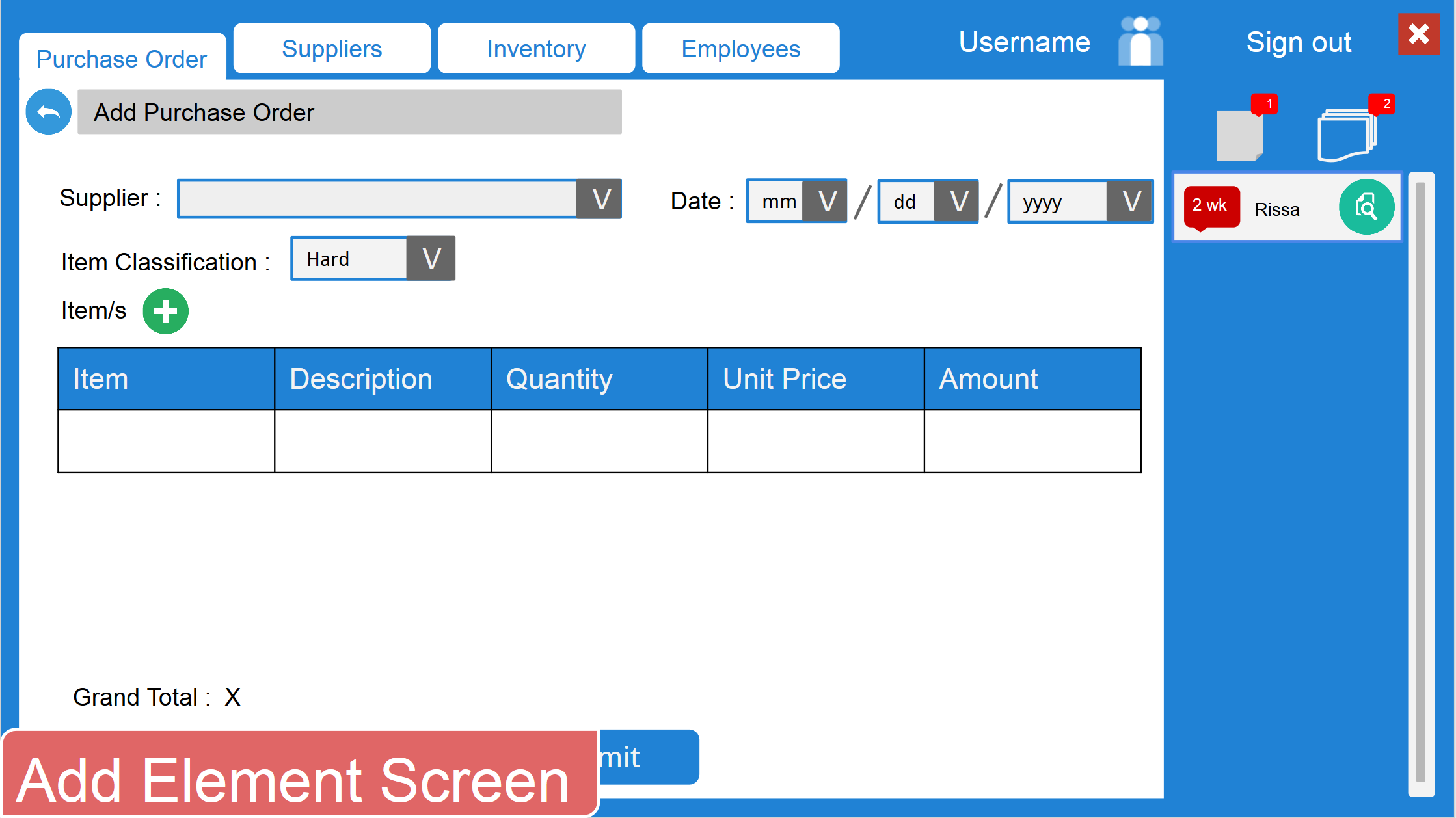
**Figure A-5. - Filter**



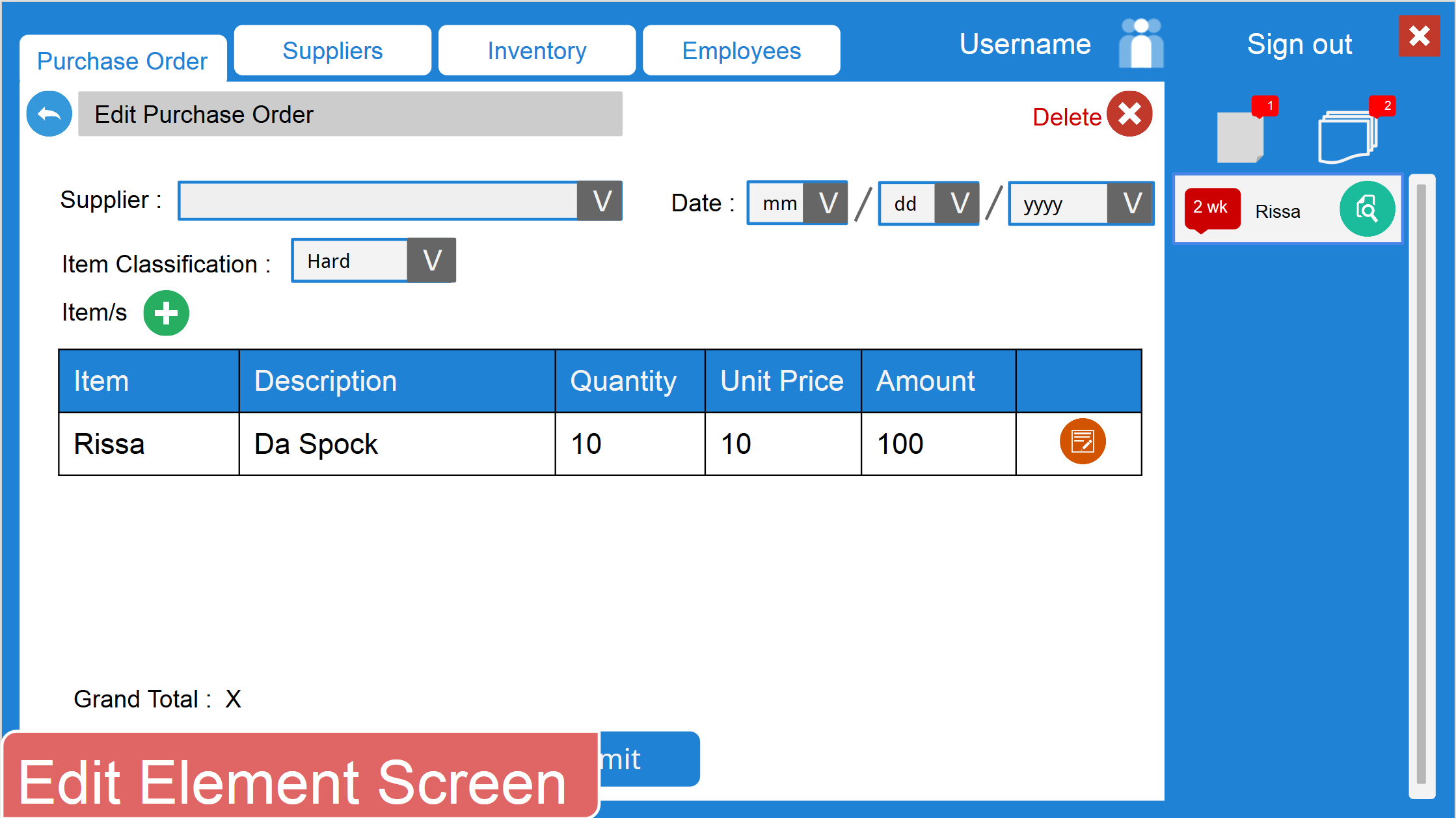
**Figure A-6. - Generate Reports Screen**



**Figure A-7. - View Element Screen**



**Figure A-8. - Add Element Screen**



**Figure A-9. - Edit Element Screen**



**Figure A-10. – Notification**

1. Messages

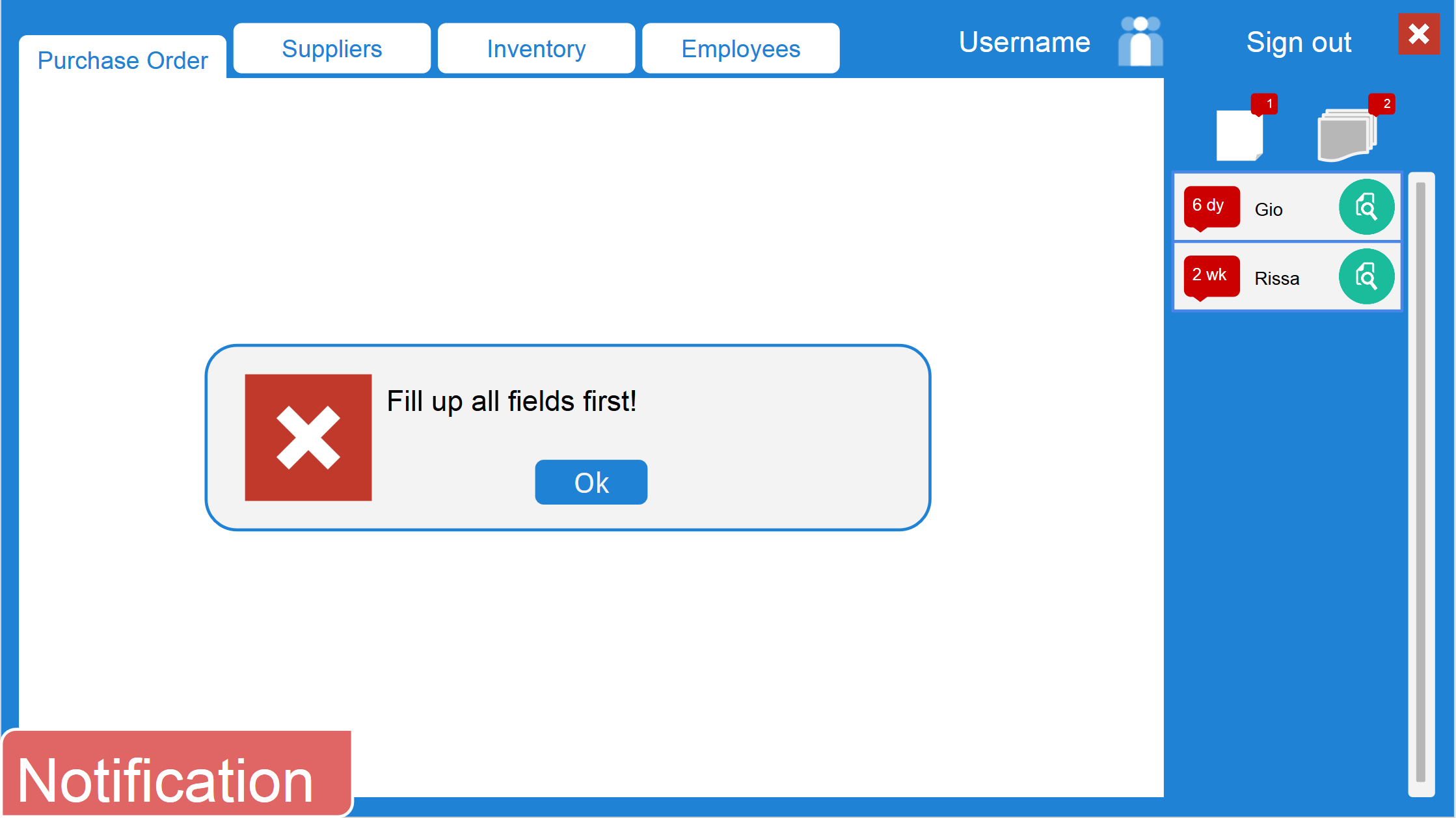


Success Information Warning Error

**Figure A-11. – Message Icons**

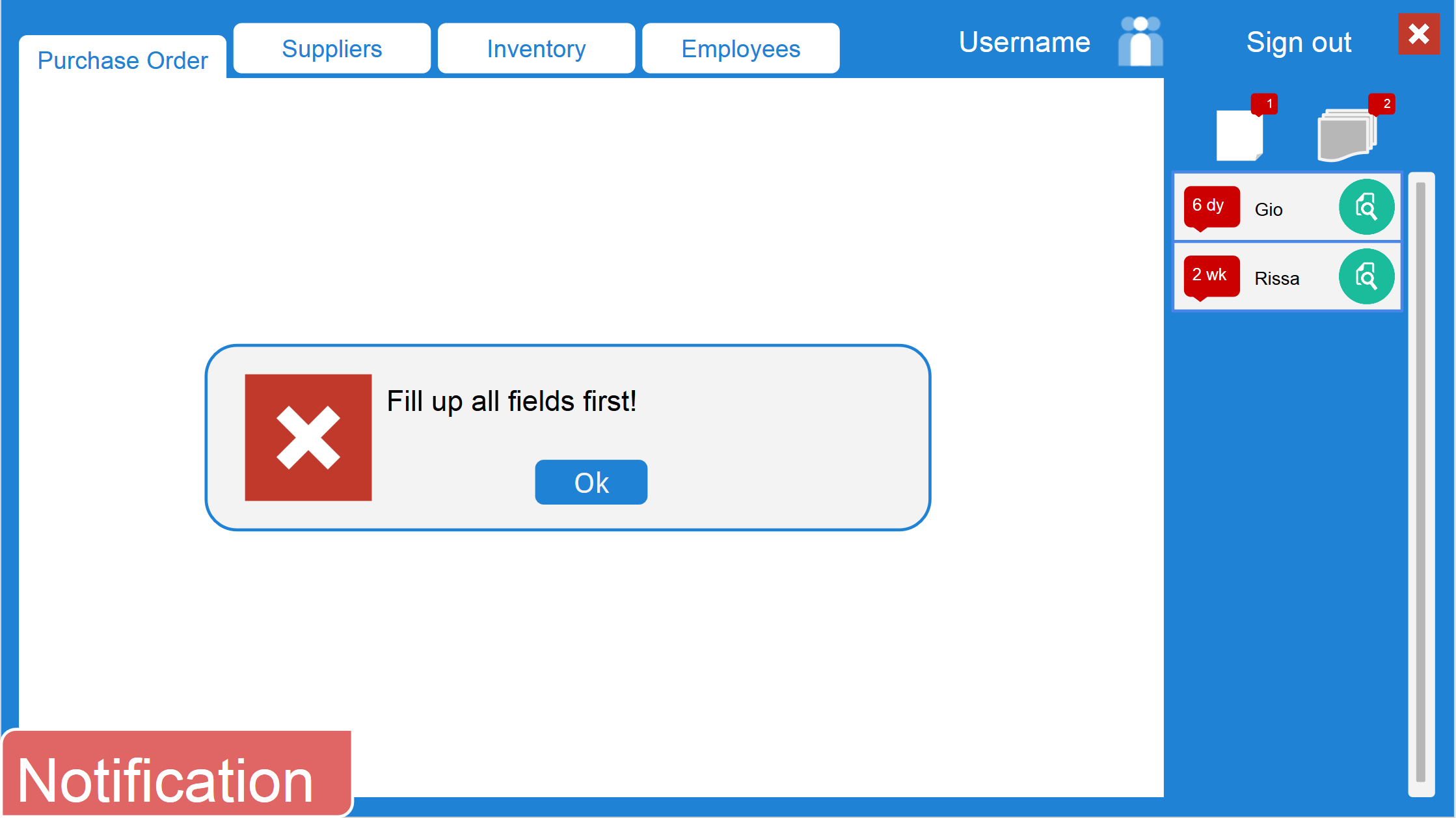
* 1. Creating
     1. Success Notification
        1. <Element> has successfully been added!
     2. Information Notification
        1. <List of details> Save new entry?
     3. Warning Notification
        1. <Element> has an existing entry already. Continue with adding new entry?
     4. Error Notification
        1. Error #<number>. System failed to save the new entry.
        2. Fill up all fields first.
        3. Invalid input for <attribute>.
  2. Reading
     1. Info Notification
        1. There are no entries to display.
        2. There are no matches to the criteria. (Filtering)
     2. Error Notification
        1. Error #<number>. System failed to retrieve data.
  3. Updating
     1. Success Notification
        1. <Element> has successfully been updated!
     2. Info Notification
        1. <List of new details> Update entry?
     3. Error Notification
        1. Error #<number>. System failed to update the entry.
        2. Fill up all fields first.
        3. Invalid input for <attribute>.
  4. Deleting
     1. Success Notification
        1. <Element> has successfully been deleted!
     2. Info Notification
        1. Are you sure you want to delete this entry?
     3. Error Notification
        1. Error #<number>. System failed to delete the entry.

1. Titles and Captions
   1. CAI-STA
      1. Sign In
   2. Suppliers
      1. List of Suppliers
      2. Add Supplier
      3. Edit Supplier Details
   3. Purchase Orders
      1. List of Purchase Orders
      2. Add Purchase Order
      3. Edit Purchase Order
      4. Generate Report
   4. Inventory
      1. List of items
      2. Add item
      3. Edit item
   5. Employees
      1. List of Employees
      2. Add Employee
      3. Edit Employee Details
2. Command Buttons
   1. Centered
3. Sort order for Lists
   1. List of Suppliers - Sort Alphabetically
   2. List of Purchase Orders - Sort by Descending Date
   3. List of Items - Sort Alphabetically
   4. List of Employees - Sort Alphabetically
4. Usability Principles
   1. Learnability
      1. Predictability
         1. The number of notifications is easily seen (Figure A-12).
         2. The type of notification selected is easily seen through a different colored icon (Figure A-12).



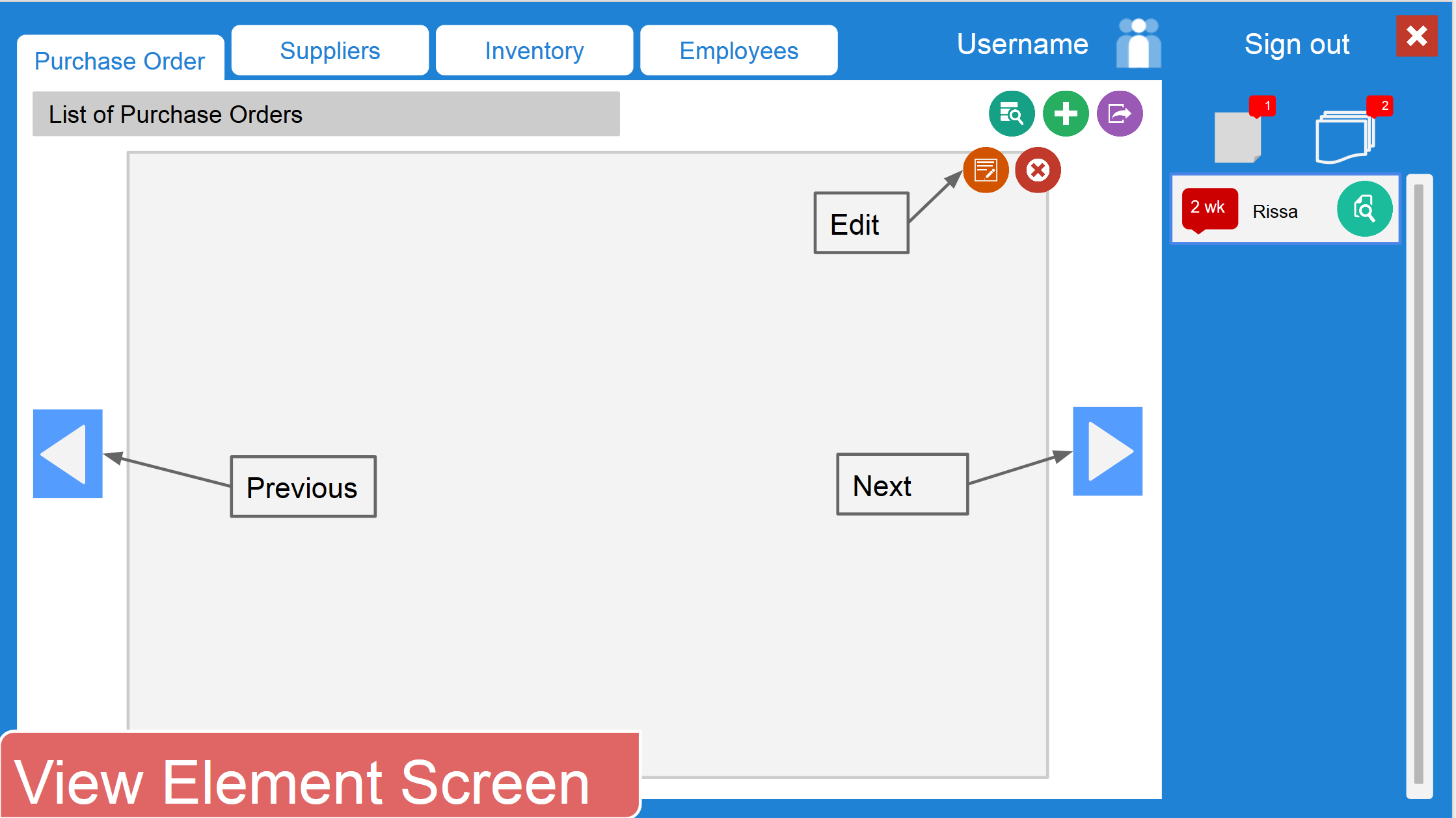
**Figure A-12. - Notifications Log**

* + 1. Synthesizability
       1. The list of elements is automatically updated upon adding and editing an item.
    2. Familiarity
       1. The presentation of the notifications is similar to Facebook’s notifications interface (Figure A-12).
       2. The tab interface is similar to web browsers (Figure A-13).



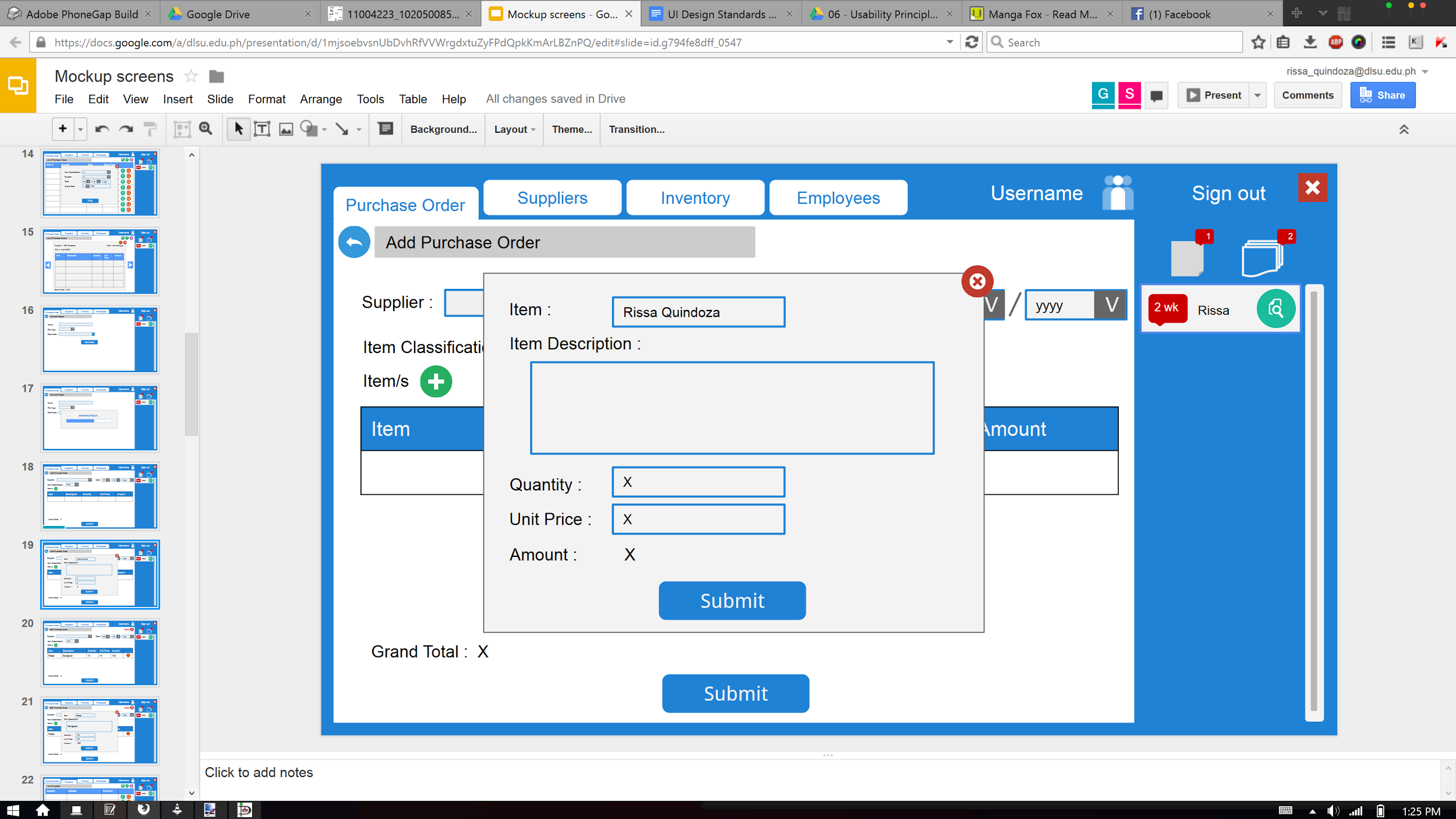
**Figure A-13. - Tab Interface**

* + 1. Generalizability
       1. The icons used are similar to Windows and Mac icons (Figure A-14).



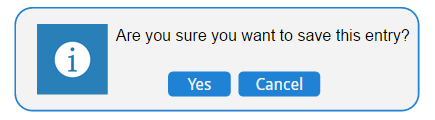
**Figure A-14. – Icons**

* + 1. Consistency
       1. The modules with similar purposes (ex. add item, add supplier, add employee) have similar layouts.
       2. Notifications have similar layouts.
  1. Flexibility
     1. Dialogue Initiative
        1. System prompts user with notifications (Figures A-16 - A-20).
     2. Task Migratability
        1. System suggests entries upon user input to reduce errors.
     3. Substitutivity
        1. System auto calculates total value for items (Figure A-15).

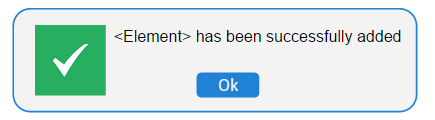


**Figure A-15. - Auto Calculate**

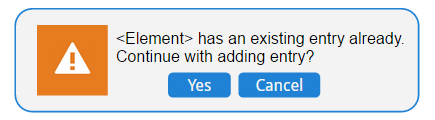
* 1. Robustness
     1. Observability
        1. User can easily change modules due to tabbed interface (Figure A-13).
        2. User can easily view notifications due to the notification log area (Figure A-12).
     2. Recoverability
        1. System prompts the user for important notifications (Figures A-16 – A20)



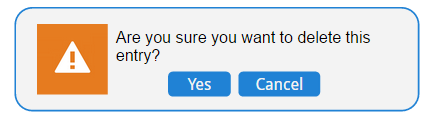
**Figure A-16. - Confirmation Notification for Submission**



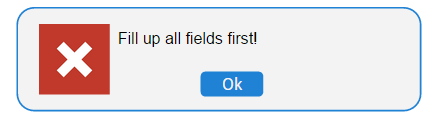
**Figure A-17. - Success Notification for Submission**



**Figure A-18. - Warning Notification for Data Duplication**

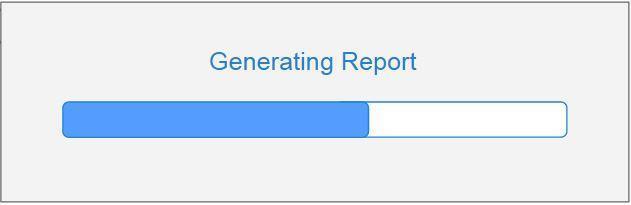


**Figure A-19. - Warning Notification for Deletion**



**Figure A-20. - Error Notification for Empty Fields**

* + 1. Responsiveness
       1. The system displays the progress of the generation of report (Figure A-21).



**Figure A-21. - Progress bar for Report Generation**

**B. Development Standards**

This section details the development standards to be followed by SystemScape to ensure a uniform process in the Software Development stage.

**B.1. Programming language, compiler, editor, debugger**

The team has decided to use Java as the programming language (particularly Java 1.8) and Eclipse Luna IDE as the standard compiler, editor and debugger for the team.

**B.2. Logic and UI Development Sequence**

The system logic and the user interface will be coded simultaneously. This is because a sub-group will be handling the user interface and another sub-group will be handling the logic. The integration of the two would be done after the user interface is finished. The logic need not be completed at the same time as the user interface.

**B.3. Top-Down or Bottom-Up Development**

Development will be done in a top-down fashion, beginning with the main module (the main entry point for the users) and then working on the sub-modules and their pertinent classes.

**B.4. Coding Standards Compliance**

When a programmer has finished coding a module, the source code is then reviewed by the Developer Head, Quality Assurance Team, and Project Manager, in that order.

**B.5. Error Handling and Security**

Each class will be tested individually (first by the developers, followed by the QA team) to determine right away if revisions are necessary. Test cases will have been made to help the programmers in validating user inputs and referential integrity.

**B.6. Programming Strategy**

Programmers will code in pairs as much as possible to help in delegating the work involved in the sub-tasks of a certain task to the two members in question.

**B.7. Version Control**

Both developers and analysts will produce outputs for the same module (e.g. adding employee logic for developers and add employee GUI for analysts) which will then be checked by the quality assurance. After syncing and testing, the output will then be pushed to the team’s code repository to be referred to as version/iteration of the software.

**B.8. Revisions and Revision Control**

GitHub will be used as the team’s code repository. The site has features which allow the handling of revisions for the existing code.

**B.9. Non-standard Language Feature Use**

The programmers could use non-standard language features if the need arises.

**B.10. Test Cases**

Test cases will be made before coding; if any additional test cases arise in the coding process, they can be added to the existing list.

**B.11. Debugging and Code Check-In**

The code should be stepped through the debugger before being uploaded to the repository. This will help in reducing bugs in the program that the programmer may have overlooked.

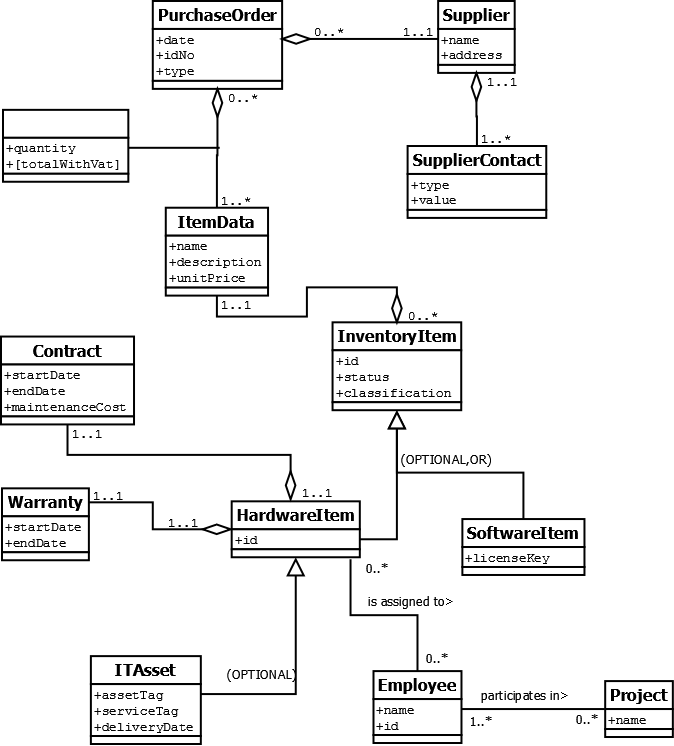
**B.12. Integration Testing**

An integration-test should be done on the module which the programmer made to ensure it works with the rest of the software.

**B.13. Code Review and Inspection**

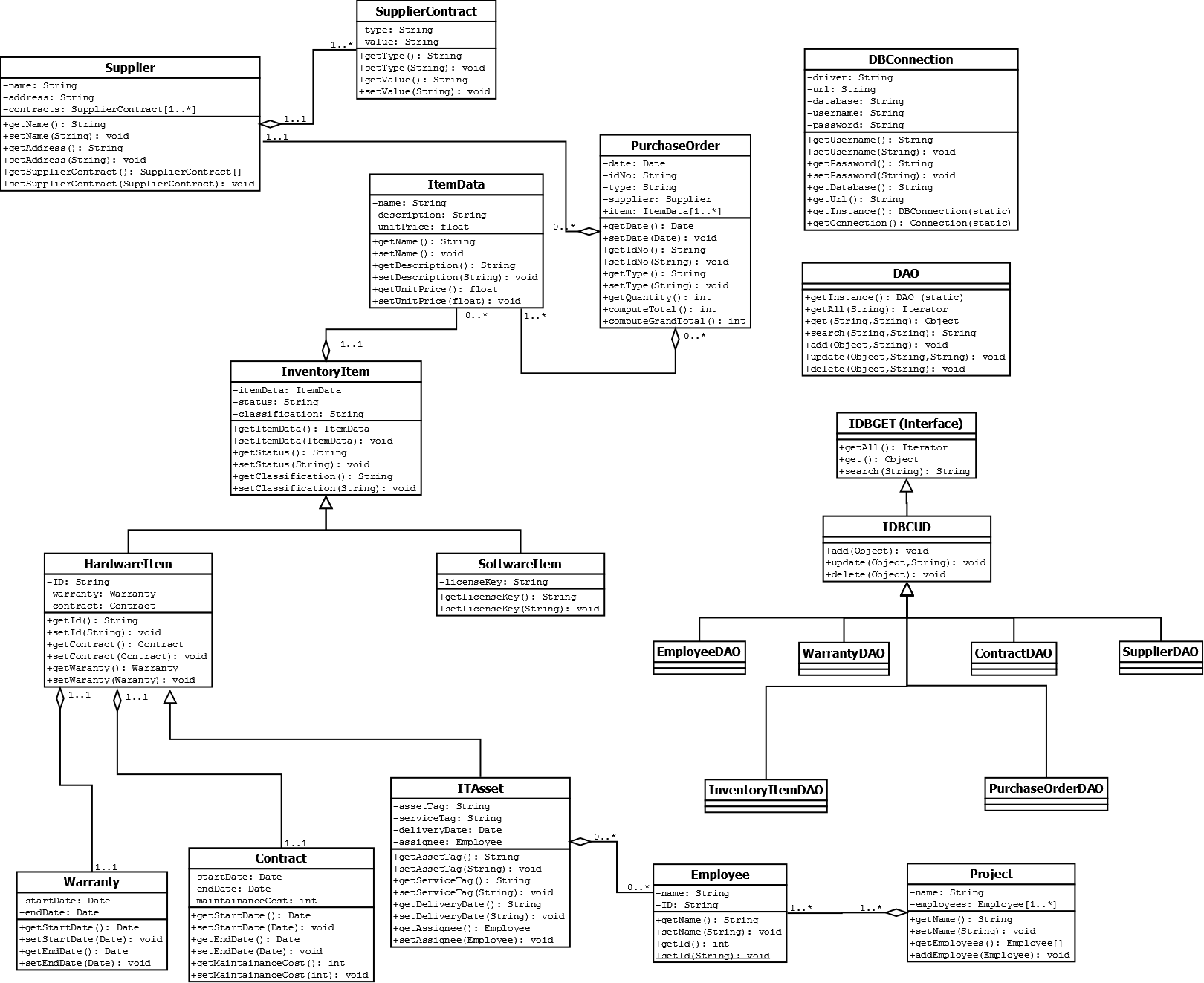
The programmers will review or inspect each other’s codes since it is important to inspect the code thoroughly to see if there are necessary revisions that need to be done; it would also be a good practice for the programmers to ensure the quality of the code.

**C. ER Diagram**



**Figure C-1. – ER Diagram for the System**

**D. Class Diagram**



**Figure D-1. – Class Diagram of the System**