**Inventory Management Application**

*Nerd Herd*

*Joshua Rozenberg, Craig Sirota, Jonnelin Marzielli Leonardo, Michael Yang*

1. **Introduction**

In a laboratory or warehouse setting, inventory regulation and tracking is of great importance to make sure that operations are not disturbed. Currently, the most common method used to track supplies is to log any and all information on paper or in an excel sheet. This makes keeping track of supplies difficult and can lead to inaccurate record keeping. Paper records can be easily lost, stolen, or modified which can lead to disruptions of operations. Our software intends to create an easy to use interface coupled with efficient data storage and insightful analysis tools. It is open source software and intended to fill the niche of small business or educational inventory management where there are currently no acceptable inventory management software solutions. The software will be open to plugins to add features to increase the functionality of the standard application.

1. **Proposed System**

Our system is an inventory management system that is configurable and is designed to be adaptable for a wide range of inventory applications. This software allows a user or group of users to efficiently and easily manage records and produce reports for any type of inventory. Data is stored in a DBMS that exists externally on a remote server. The user will also be able to specify their own database server for data storage if they choose. The main system functionalities include: configuring the system, scanning and viewing items, searching and sorting items, editing item information, adding and deleting items, and generating reports. The system shall have 3 roles that permit the various functionalities. The first role is the Admin who has the ability to read, add, edit, delete and perform admin tasks. Admin tasks include granting roles, generating reports, and administering the system. The next role is Employee; and an employee has permissions to read, add, edit and delete. Finally, there is the Customer; the customer role has permission only to read or view items

* 1. *Functional Requirements*

|  |  |
| --- | --- |
| IN - 001 | Users should be able to scan an item and find information about that item, specified by the system administrators. |
| IN - 002 | Users should be allowed to modify the amount of an item in a list. |
| IN - 003 | Users should be allowed to manually add new items. |
| IN - 004 | Users should be allowed to add categories to organize the items by. |
| IN - 005 | Users should be able to filter items to make looking for items easier. |

|  |  |
| --- | --- |
| OT - 001 | The application shall return all the items data after an item is scanned. |
| OT - 002 | The system shall send inventory reports to all system administrators on a specified schedule. |
| OT - 003 | The application shall prompt the user if the scanned item is not found in  the database. |
| OT - 004 | A notification should be sent out to the appropriate individuals regarding a change. |
| OT - 005 | Show items sorted in alphabetical order and based on categories and filter settings. |

|  |  |
| --- | --- |
| PT - 001 | The application shall allow users to scan existing items and retrieve. |
| PT - 002 | The application shall allow the admin to generate basic statistical reports on the inventory. |
| PT - 003 | The system should be configurable by the admin. (custom categories and fields for different applications) |
| PT - 004 | The application shall allow the admin to generate a sales analysis to predict future inventory requirements. |
| PT - 005 | The application shall allow all users to search, sort, and filter inventory results. |
| PT - 006 | The Application shall allow admins to assign user roles |

* 1. *Non-Functional Requirements*

|  |  |
| --- | --- |
| IN - 101 | Users should not be allowed to access supply listings that are outside their level of permissions. |
| IN - 102 | Users should not be allowed to modify the quantity of an item without correct permissions. |
| IN - 103 | Items being added should not be in the system already. |
| IN - 104 | Item addition should only be done when all required fields are filled out. |
| IN - 105 | Modifying information in the database should be logged. |

|  |  |
| --- | --- |
| OT - 101 | Data that is output should always be complete, correct, and consistent with the database. |
| OT - 102 | The system should catch and handle any data errors or SQL exceptions on output. |
| OT - 103 | Document output and user requesting output and store information in a log. |
| OT - 104 | Output data should always be exactly consistent with the data request or an error shall be reported. |

|  |  |
| --- | --- |
| PT - 101 | Access to the data should be exactly consistent with the defined permissions. |
| PT - 102 | The system shall monitor storage capacity before each save and send alerts if storage is 85% full. |
| PT - 103 | The software shall display the UI responsively between different android devices. |
| PT - 104 | The system shall use multi-threading to separate background tasks and UI tasks. |
| PT - 105 | The system shall not take longer than 10 seconds to retrieve and display proper information when network connection is not an issue. |
| PT - 106 | The system shall store all data in a DBMS on an external server. |
| PT - 107 | The application shall be open source and available at no cost. |
| PT - 108 | Multiple users should be able to access the database at the same time. |

* 1. *System Models*

**Users/Roles/Actors**

1. Customer
   1. has read only permission; can see inventory but not add, edit, delete or any admin tasks
2. Employee
   1. has permission to read, add, edit, delete; no admin tasks allowed
3. Manager
   1. has permission to read, add, edit, delete, and to perform all admin tasks

**General Use Cases**

1. Retrieve item information (scan)
   1. All users
2. Edit item information
   1. Employee
   2. Manager
3. Add item
   1. Employee
   2. Manager
4. Search/sort/filter inventory
   1. All users
5. Application configuration (fields, categories)
   1. Manager
6. System administration (permissions, users)
   1. Manager
7. Generate basic reports
   1. Manager
8. Generate sales analysis
   1. Manager
      1. Use Cases

|  |  |
| --- | --- |
| *Use Case Name* | Generate Sales Report |
| *Participating Actors* | Manager |
| *Flow of Events* | 1. User clicks on the sales report button 2. System displays a dialog 3. User enters their desired options 4. User clicks the generate button 5. System generates and displays the report 6. User exports the report or quits |
| *Entry Condition* | User has Admin permissions |
| *Exit Conditions* | User clicks the export button, cancels, or closes the window |
| *Quality Requirements* | generating the report should not take more than 1 min on a good network connection and server is working correctly |

|  |  |
| --- | --- |
| *Use Case Name* | Generate Statistics Report |
| *Participating Actors* | Manager |
| *Flow of Events* | 1. User clicks on the basic statistics button 2. System displays a dialog 3. User enters their desired options 4. User clicks the generate button 5. System generates and displays the report 6. User exports the report or quits |
| *Entry Condition* | User has Admin permissions |
| *Exit Conditions* | User clicks the export button, cancels, or closes the window |
| *Quality Requirements* | generating the report should not take more than 30 sec on a good network connection and server is working correctly |

|  |  |
| --- | --- |
| *Use Case Name* | Modifying Employee Access |
| *Participating Actors* | Manager |
| *Flow of Events* | 1. User clicks on the system settings button 2. System opens the system settings window 3. System retrieves the stored configuration file and opens it 4. User accesses employee list 5. User is presented with options to add, remove, and edit employee permission. 6. User makes the desired change(s) 7. User clicks save 8. System saves the old configuration as a backup 9. System validates and then applies the new settings 10. System attempts to reload the configuration and reverts to the backup if there is a failure |
| *Entry Condition* | User has Admin permissions |
| *Exit Conditions* | User clicks the save button, cancels, or closes the window |
| *Quality Requirements* | The system settings should never be able to be corrupted by user modification.  All exceptions should be handled and reported to the user. |

|  |  |
| --- | --- |
| *Use Case Name* | Add Item |
| *Participating Actors* | Employee or Manager |
| *Flow of Events* | 1. User clicks the add item button 2. System responds by showing an input dialog 3. User inputs the desired information 4. System responds by notifying server an item should be added 5. System responds by adding the new item |
| *Entry Condition* | User has permission to add an item |
| *Exit Conditions* | User clicks the save new item button, cancels, or closes the window |
| *Quality Requirements* | Adding a new item should not take more than 30 seconds on a good network connection and server is working correctly. |

|  |  |
| --- | --- |
| *Use Case Name* | Edit Item |
| *Participating Actors* | Employee or Manager |
| *Flow of Events* | 1. User clicks on the edit button 2. System responds by opening the edit screen 3. User changes the name, QR/bar code, category, or quantity in the form 4. User clicks the save button 5. System responds by taking values in form and sending it to server to be saved 6. System responds by notifying user changes have been saved 7. System responds by notifying other users regarding change 8. System responds by returning to the previous screen |
| *Entry Condition* | User has permission to edit  Item exist in database |
| *Exit Conditions* | User clicks the save button, cancels, or closes the window |
| *Quality Requirements* | Saving changes should not take more than 30 seconds on a good network connection and server is working correctly. |

|  |  |
| --- | --- |
| *Use Case Name* | Delete Item |
| *Participating Actors* | Employee or Manager |
| *Flow of Events* | 1. User clicks the delete item button 2. System responds by showing a confirmation dialog 3. User clicks the confirm delete button 4. System responds by notifying server an item should be deleted 5. System responds by deleting item from list 6. System responds by notifying other users regarding change |
| *Entry Condition* | User has access to delete item  Item exists in database |
| *Exit Conditions* | User clicks the confirm delete button, cancels, or closes the window |
| *Quality Requirements* | Deleting item should not take more than 30 seconds on a good network connection and server is working correctly  Item should be deleted from the list in device within 5 seconds |

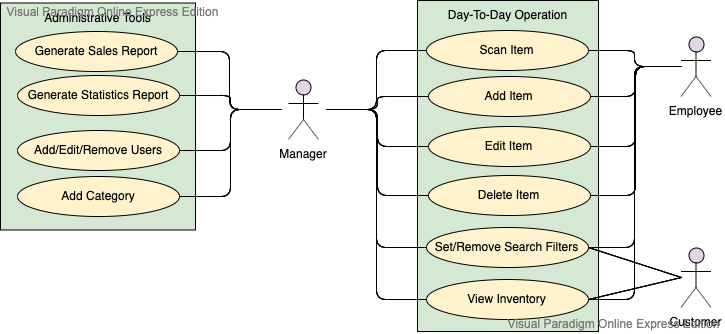
|  |  |
| --- | --- |
| *Use Case Name* | Add Category |
| *Participating Actors* | Manager |
| *Flow of Events* | 1. User clicks on add category button 2. System responds by switching to add category screen 3. User fills out the form including the name of the category 4. User clicks on the submit button 5. System responds by alerting server a new category should be added 6. System responds by notifying user a new category has been successfully added 7. System responds by returning to previous screen |
| *Entry Condition* | User has access to add category |
| *Exit Conditions* | Category does not already exist  User clicks on submit button, cancels, or closes the window |
| *Quality Requirements* | Saving new category should not take more than 30 seconds on a good network connection and server is working correctly |

|  |  |
| --- | --- |
| *Use Case Name* | Filter Items |
| *Participating Actors* | Any user |
| *Flow of Events* | 1. User clicks on filter item list button 2. System responds by showing filter view 3. User selects to filter by category or quantity range 4. User selects filter button 5. System responds by filtering list according to users filter selections |
| *Entry Condition* | User exists |
| *Exit Conditions* | User clicks on filter button, cancels, or closes the window |
| *Quality Requirements* | System should filter list within 5 seconds |

|  |  |
| --- | --- |
| *Use Case Name* | Clear Filters |
| *Participating Actors* | Any user |
| *Flow of Events* | 1. User clicks on clear filter button 2. System responds by removing filter selections 3. System responds by displaying list unfiltered |
| *Entry Condition* | User exists |
| *Exit Conditions* | User clicks on clear filter button, cancels, or closes the window |
| *Quality Requirements* | System should clear filters within 5 seconds |

|  |  |
| --- | --- |
| *Use Case Name* | Scan Item |
| *Participating Actors* | Any User |
| *Flow of Events* | 1. User clicks on scan item button 2. System responds by opening the camera 3. User takes a picture of the QR/bar code 4. System responds by searching for item with the same code 5. System responds by opening a screen with the item’s information, including name, category, QR/bar code, and quantity |
| *Entry Condition* | User exists |
| *Exit Conditions* | User takes a picture, cancels, or closes the window |
| *Quality Requirements* | System should not take more than 30 seconds to retrieve correct item with a good network connection and the server is working properly |

* + 1. Use Case Diagrams

**

1. **Glossary**

Filter: A way to limit what items are being displayed

Item: Any entity the system is in charge of keeping track of

API: Application Programming Interface

1. **Reference**

Barcode API: <https://developers.google.com/vision/android/barcodes-overview>

Database API: <https://www.python.org/dev/peps/pep-0249/>

<https://www.drupal.org/docs/8/api/database-api/database-api-overview>