

**Bullseye Sporting Goods**

**Bullseye Inventory Management System**

**(BIMS)**

**2025 Initiative**

**Individual Software Development Project**

**Specifications**

**Document Changes**

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Notes** |
| 1.0 | Chris London | First Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

**Contents**

[**INTRODUCTION** 5](#_Toc160614906)

[**EXPECTATIONS** 6](#_Toc160614907)

[**Security and Permissions** 7](#_Toc160614908)

[**Coding Expectations** 8](#_Toc160614909)

[**Coding Standards** 8](#_Toc160614910)

[**Usability Standards** 8](#_Toc160614911)

[**Tab Order & Logical Design** 9](#_Toc160614912)

[**Input Masks** 9](#_Toc160614913)

[**Data Validation** 10](#_Toc160614914)

[**Searches** 10](#_Toc160614915)

[**Defaults** 11](#_Toc160614916)

[**OVERVIEW** 12](#_Toc160614917)

[**EXISTING SYSTEM** 13](#_Toc160614918)

[**Standard Order delivery days** 14](#_Toc160614919)

[**KEY PERSONNEL** 15](#_Toc160614920)

[**Eduardo Concepcion** **Regional Manager** 15](#_Toc160614921)

[**Monica Munoz** **Finance Manager** 15](#_Toc160614922)

[**Chris Patstone** **Warehouse Foreman** 15](#_Toc160614923)

[**Jose Perez**  **Store Manager** 15](#_Toc160614924)

[**Charles Norris** **Delivery Driver** 15](#_Toc160614925)

[**EPICS** 16](#_Toc160614926)

[**SPRINT 1** 17](#_Toc160614927)

[**Admin Setup** 17](#_Toc160614928)

[**Database Confirmation** 17](#_Toc160614929)

[**Dashboard** 18](#_Toc160614930)

[**Login** 19](#_Toc160614931)

[**Logout** 20](#_Toc160614932)

[**Add User** 21](#_Toc160614933)

[**Edit User** 22](#_Toc160614934)

[**Delete User** 23](#_Toc160614935)

[**Read User** 24](#_Toc160614936)

[**Set User Permissions** 25](#_Toc160614937)

[**Change Password** 26](#_Toc160614938)

[**Edit Item** 27](#_Toc160614939)

[**SPRINT 2** 28](#_Toc160614940)

[**Audit Activity** 28](#_Toc160614941)

[**Move Inventory** 28](#_Toc160614942)

[**Create Store Order** 29](#_Toc160614943)

[**Create Emergency Order** 31](#_Toc160614944)

[**Receive Store Order** 33](#_Toc160614945)

[**Fulfil Store Order** 34](#_Toc160614946)

[**Add Item to Backorder** 35](#_Toc160614947)

[**View Store Order** 36](#_Toc160614948)

[**View Site** 37](#_Toc160614949)

[**Add Site** 37](#_Toc160614950)

[**Edit Site** 39](#_Toc160614951)

[**Edit Inventory** 40](#_Toc160614952)

[**SPRINT 3** 41](#_Toc160614953)

[**Check Delivery** 41](#_Toc160614954)

[**Pickup Store Order** 41](#_Toc160614955)

[**Transport Store Order** 42](#_Toc160614956)

[**Deliver Store Order** 43](#_Toc160614957)

[**Accept Store Order** 44](#_Toc160614958)

[**Place Online Order** 45](#_Toc160614959)

[**View Online Order** 46](#_Toc160614960)

[**Prepare Online Order** 46](#_Toc160614961)

[**Receive Online Order** 47](#_Toc160614962)

[**Modify Record** 48](#_Toc160614963)

[**Add Supplier** 49](#_Toc160614964)

[**Edit Supplier** 50](#_Toc160614965)

[**SPRINT 4** 52](#_Toc160614966)

[**Create Loss** 52](#_Toc160614967)

[**Process Return** 52](#_Toc160614968)

[**Add New Product** 53](#_Toc160614969)

[**Edit Product** 53](#_Toc160614970)

[**Create Supplier Order** 54](#_Toc160614971)

[**Create Reports** 54](#_Toc160614972)

# **INTRODUCTION**

Welcome to the show! 😊

This is your introduction to the big tamale, the big kahuna, the end game – the capstone project!

This capstone project is based on a case study in which you must build a fully functional software system from a combination of given set of specifications and your own creativity and knowledge/skills.

This course is an opportunity for students to gain practical experience in system development. Students apply systems design theory and computer programming skills to complete a small systems development project under the supervision of an instructor.

Student progress is closely monitored by the instructor(s) using intermittent technical reviews and manual submissions. Students program, test, document and deliver a realistic, small-scale system with components written in a variety of languages against an approved Relational Database Management System (RDMS). Emphasis is placed upon ***individual initiative, resourcefulness, and self-discipline*** to build and implement this project from start to finish.

This document contains the guidelines for your project. It is not all-inclusive, there is plenty of room for you to add your own functionality and creativity, but the overall expectation is that your final project will perform the required features as designed with the goal of providing the end users a properly working system.

To ensure we can assess - and you can demonstrate - the range of your abilities, ***you will create a combination of a desktop and web-based app***, using the **provided MySQL database (bullseyedb2023\_1.0).** If this database script needs to be tweaked along the way, the version will be updated at the end and you will be notified (bullseyedb2023\_1.1, bullseyedb2023\_1.2, etc.).

You may **add new tables, fields, triggers, stored procedures, etc.,** to the database to handle any functionality you may wish to add, but **you may not alter or drop features in existing tables *or data therein***.

If you add new tables/triggers/etc., you must create them using a separate .sql file to run ***after running the file provided***, as we will be running the provided file prior to all sprint reviews to ensure we have a valid database and my test data runs as expected

# **EXPECTATIONS**

* Discipline, product ownership, professionalism, dedication to the task at hand
* Efficient coding practices
* Proper commenting
* All code and documentation will be stored in a private GitHub library, which will be shared ONLY with your instructors (via email invite)
* Weekly log in which you journal your progress
* Week 1 is prep and setup, then you have four (4) sprints of three (3) weeks duration each (Sprint 2 is 4 weeks because it encompasses Student Success Week). Each sprint has a series of features/requirements to be completed, as listed in this document
* **Assessments:**

Item % Due

* + Sprint 1 20% After 4 weeks
  + Sprint 2 25% After 8 weeks
  + Sprint 3 25% After 11 weeks
  + Sprint 4 25% After 14 weeks
  + Journals 5% Minimum one (1) **substantial** journal entry/week
* **Formal Sprint Planning** sessions will occur the first scheduled class at the start of each sprint (**attendance is mandatory**).
* Weekly demonstrations of your progress will occur during scheduled class times and formal evaluation of your work can be incremental and you can continue to improve upon it as the sprint progresses. If you wait until the end for a final review, that is your only assessment and there will be no opportunity to improve
* **Formal Sprint Reviews** will occur at the end of each sprint for those who have not had their work graded incrementally (**attendance is mandatory**)
* Grades are based on functionality based on the given specifications and accepted coding/UX standards. ***If you are unsure of what is meant in a specification, ASK.* Getting the specification wrong because you did not ask is not an excuse**
* Penalties are given for inefficient code, commenting, poor UX / design, crashes during reviews. At this point in your career, avoiding these is expected. You will not receive credit for doing it, but you will be penalized for not following specs, not writing efficient code, not commenting, not following good UX / design standards, and crashes due to improper testing
* Every sprint is REQUIRED to successfully complete this course (i.e. you cannot decide not to submit a sprint and be successful). You cannot progress to the next sprint until the current/previous one has been reviewed
* This is an **INDIVIDUAL** project. There will be **NO SHARING CODE**. You may only share “***concepts***” with your classmates in an effort to help them out of a jam.
  + (Example: Someone having trouble connecting to a DB with a data table, you could help by giving them advice like “I solved that by using this library \_\_\_\_\_\_\_\_\_\_” or “Did you remember to do this \_\_\_”.

## **Security and Permissions**

The Bullseye Inventory Management System (BIMS) has role-based permissions. To access a particular aspect of the system, the user logged in must have the appropriate role. Each feature/functionality will have the appropriate role(s) listed with the other specifications.

**System Role Permissions:**

* Regional Manager
* Financial Manager
* Store Manager
* Warehouse Manager
* Delivery
* Warehouse Employee
* Administrator

## **Coding Expectations**

The Bullseye Inventory Management System (BIMS) client expects nothing but the best User Experience. To facilitate this, we will use all appropriate User Experience methodologies you have learned at NBCC to make this a smooth experience.

### **Coding Standards**

All variable names will be self-describing, as learned while at NBCC.

**Examples:** customerID, employeeID, txnID, firstName, lastName, status, notes, active, etc.

### **Usability Standards**

All forms/reports should follow best practices for industry standard UX design and usability.

Examples include, but are not limited to:

* User should see their username and what site/location they are associated with on **every form/report**
* Corporate logo will be displayed in the top right or left corner on **every form/report**
* Maintain same UX design/look & feel, color, etc., on **every form/report**
* All messages (error or other) will be context-sensitive, intelligent, self-describing, and contain no ambiguity
* When showing lists and data, include descriptions and not simply id values (i.e. make it intelligent and easy to use) on **every form/report**
* Maintain a context-sensitive help that can be activated by selecting F1 and a Help menu item on **every form/report**
* Enable search capabilities that allow a user to search for a specific record by id or description on **every form/report**
* Data grids/tables/lists should not populate automatically upon form opening (very bad for efficiency). They should have a ‘Refresh’ button or function that launches it to prevent wasted resources connecting to a DB on **every form/report** that has a grid/list/table
* Whenever there is a selection/choice to make involving <=15 choices, use a dropdown combo box
* Whenever there is a selection of >15 choices, use a list box or add a search feature instead of a combo box
* Whenever dates/times are required, use datetimepickers to eliminate data entry errors/miscommunications
* Datetimepickers should default to current date. If using multiple with a range (i.e. from -> to ranges), the second datetimepicker should default to the same datetime or one that makes sense ***after*** the first one as soon as the first date once the first is set
* Using a dashboard with tabs or similar that manages overall look and feel rather than creating many different forms that would need to be lined up for look and feel is much easier/more efficient

### **Tab Order & Logical Design**

All input fields should be in proper logical order and the tab order should follow that flow.

### **Input Masks**

All input items which should logically require a mask to validate input ***must be set up with an input mask that ensures only valid data can be entered and only the alphanumeric is captured in the database***. No storing of spaces, braces, hyphens, etc.

**Examples:** postal code, phone number, email address

A close-up of a number

Description automatically generated with low confidenceGraphical user interface, text, application, chat or text message

Description automatically generated **Stored in DB Displayed on form/screen**

## **Data Validation**

When possible, all input data that would not reasonably change should be selected from a combo box, list, radio button, checkbox or similar.

**Examples:** Province, Country, day of week, site, supplier, siteType, txnType, txnStatus, posn, etc.

Diagram

Description automatically generated

## **Searches**

All searches will be dynamic, This means that the search starts immediately and auto-completes as the user enters more info.

Graphical user interface, application

Description automatically generated**Example:**

## **Defaults**

Whenever there is an obvious default selection (i.e. selected more than 50% of the time), that item should be the default value in any input field

**Examples:** Province: NB should be default, Country: Canada should be default, Active: should be True or 1 by default

Text

Description automatically generated

# A shopping cart full of toys Description automatically generated**OVERVIEW**

Bullseye is a growing Sporting Goods retail chain based in Atlantic Canada that wants to expand into the rest of Canada. Starting with a single retail location in Saint John, NB, they have expanded slowly across New Brunswick with a plan to secure a foothold in the rest of Atlantic Canada before expanding to the rest of the country.

Bullseye has existed as a business for the past twenty (20) years. During the past 2 years Bullseye has purchased a string of seven (7) different, independent retail operations in New Brunswick in order to build their presence.

[](https://www.google.ca/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwityOSc4v3OAhWRsh4KHYjPDAIQjRwIBw&url=https://www.colourbox.com/image/james-hardy-altopress-maxppp-towering-shelving-racks-housing-stacks-of-boxes-in-warehouse-ningbo-zhejiang-province-china-image-1619029&psig=AFQjCNE4MV7ZzmoKfr_6k8MhfRY7lGKpfg&ust=1473355267701035)It has also established regional warehouse facilities to handle centralized purchasing and distribution. This places the product closer to the retail stores while allowing Bullseye to take advantage of bulk discounts from major suppliers.

|  |  |
| --- | --- |
| ClothingA store with many clothes and shoes  Description automatically generated with medium confidence | FootwearA collection of colorful shoes  Description automatically generated |
| AccessoriesA person holding a skipping rope  Description automatically generatedA person running on a dirt path  Description automatically generated | Equipment A group of sports equipment  Description automatically generated |
| WellnessA person and person on exercise bikes  Description automatically generated | SeasonalA person walking on a golf course  Description automatically generated |

The largest retail location, in Saint John, NB, offers a full-service sporting goods store experience, including everything you would expect to find in a modern sporting goods chain (Clothing, Footwear, Accessories, Equipment, Wellness and Seasonal). Since it is the largest and first site it is also the location of the regional warehouse for Southern NB.

Bullseye is under pressure to compete with other, larger retail giants (Cleves, Mark’s, and Walmart), and watched with interest what happened when American retail competitor Target tried unsuccessfully to enter the Canadian market in a hurry. Bullseye wants to avoid the catastrophic results they witnessed from Target's attempt by ensuring they do a better job of analyzing their needs and having a solid, working system in place that integrates all seven retail locations into a single system for purchasing and distribution of merchandise.

A group of people in a meeting

Description automatically generatedThe increase in complexity and amount of new business locations and regional warehouses as a result of the expansion has resulted in growing pains. Integration of multiple retail operations has become more and more complex. A process that is relatively simple to maintain for a single location has grown in complexity to an almost insurmountable level. Things are so bad that many problems are apparent. Each retail operation had its own system for managing orders, inventory, sales, and reporting. Now local warehouses **(STOREROOMS IN LOCAL STORES)** have been created to handle storage of products. Unfortunately, this resulted in a significant amount of work to collect and report on what is being done at the various locations not to mention sourcing inventory and delivering products using seven different order formats. One store even has started using their own inventory ID numbers based on suppliers’ values to try to streamline orders. There are lost savings on bulk order purchasing and redundancy of work, just to name a few potential areas of improvement.

Also, the old systems are outdated, patched, broken, isolated from each other, and often inaccurate. Bullseye has decided to formally investigate the existing processes, improve and correct them, and then implement a new system to automate and streamline as much as possible.

Your consulting firm was invited to undertake a study, with the goal of submitting a design and proposal for a new system. This document contains the specifications for that new system.

# **EXISTING SYSTEM**

The current Inventory Control System (ICS) used by the largest retail store is an old, Windows 3.1-based application that has been in place for almost 20 years. In all aspects it is outdated and is in such a state that the reality is that many people have reverted to using manual or mental processes. Nothing would be gained from exploring the features of this old system. Suffice it to say that the retail end is functioning fairly well but the warehouse is barely functioning at all. Store managers have started calling the warehouse manager directly to communicate ‘Emergency Orders’ to keep the shelves stocked.

[A person and person in a factory

Description automatically generated](http://www.superstock.co.uk/stock-photos-images/1439R-1025031)

Bullseye is interested in a new, custom solution built from the ground up (including applications, hardware and communications), and does not wish to migrate anything from the old system(s).

Currently the store managers handle the decision of which items will be ordered and the amounts of each order based upon the sales and the amount of space available on the shelves. They have limited storage space in the stores’ warehouse area so stock should go directly onto the shelves whenever possible. When absolutely necessary excess stock (what won’t fit on the shelf) is placed on marked shelves ‘out back’ in the storage area and the amounts and location are recorded in the inventory system at the store level.

A semi truck with a container on the side

Description automatically generatedOrders are placed weekly with each store having been assigned a day (Monday to Friday) on which they place their orders. Spreading the work evenly throughout the week for the warehouse operations. At least that was the plan…

A person on the phone and using a computer

Description automatically generatedSince ordering has become complex the store managers have started using Emergency Orders to keep stock on the shelves. Although this has helped the retail end of the business it is straining the warehouse system to the breaking point.

### **Standard Order delivery days**

MONDAY: Saint John

TUESDAY: Moncton, Dieppe (Same truck, 2 orders on 1 delivery)

WEDNESDAY: Oromocto, Fredericton (Same truck, 2 orders on 1 delivery)

THURSDAY: Miramichi

FRIDAY: Sussex

# **KEY PERSONNEL**

## **Eduardo Concepcion Regional Manager**

Responsible for the overall management of all Bullseye operations in Atlantic Canada. Eduardo has the authority to manage all aspects of the Bullseye operations as they relate to the Canadian expansion. Eduardo is the project champion for this project.

## **Monica Munoz Finance Manager**

Monica is responsible for the finances for all Bullseye operations in Canada. Monica reports to Eduardo.

## **Chris Patstone Warehouse Foreman**

Chris is the “interface” between the suppliers, the warehouse, and the individual retail locations. Chris is in charge of keeping inventory moving to where it is needed. Chris reports to Monica Munoz.

## **Jose Perez Store Manager**

One of seven store managers for Bullseye in Canada, Jose is responsible for the daily management of the largest retail location, located in Saint John, NB. Jose reports to Eduardo Concepcion.

## **Charles Norris Delivery Driver**

Delivery drivers are unionized employees of Acadia Trucking Ltd., and contracted to Bullseye. Technically, all drivers report to Chris Patstone, but they also report to a supervisor at Acadia Trucking Ltd.

# **EPICS**

|  |  |
| --- | --- |
| **Admin**   * Admin Setup * Database Confirmation * DAD * Data Dictionary * Dashboard * Audit Activity * Login * Logout * Add User * Edit User * Delete User * Read User * Set User Permissions * Change Password * Edit Item   **Warehouse**   * Add New Product * Edit Product * Create Supplier Order * Edit Inventory * Add Supplier * Edit Supplier * Edit Item * Add Item to Backorder | **Store**   * Create Store Order * Create Emergency Order * Accept Store Order * Receive Store Order * Prepare Store Order * Fulfil Store Order * Create Loss * Process Return   **Delivery**   * Pickup Store Order * Transport Store Order * Deliver Store Order * Check Delivery   **Online**   * Place Online Order * View Online Order * Prepare Online Order * Receive Online Order |

# **SPRINT 1**

|  |  |
| --- | --- |
| Admin Setup  Database Confirmation  DAD  Data Dictionary  Dashboard  Audit Activity  Login  Logout | Add User  Edit User  Delete User  Read User  Set User Permissions  Change Password  Edit Item |

## **Admin Setup**

**Description: Setup the** Cone of the following for tracking your activities:

Trello

Jira

**Acceptance Criteria:**

* Properly completed activity tracking is setup and populated for the project

**Tasks:**

* Create Trello (or Jira) project for this course. Project name is: **isdp2025\_yourfirstname\_yourlastname**
* Create a GitHub repository for this project. Project name is: **isdp2025\_yourfirstname\_yourlastname**
* Update the Trello or Jira project with the associated user stories and tasks
* Invite your instructor(s) via email to your Trello/Jira project and your GitHub project. You can do this inside your GitHub repository by going to *Settings > Collaborators > Add by email*
* Your GitHub repository will house ALL files associated with this project, including this document and any document(s) you create, .sql files, source code, web pages, config files, notes, etc. This is your backup. “My hard drive crashed” or “My laptop died” are not excuses for losing your work. Keep your repository updated regularly. This is a work/life lesson that will serve you well in this industry.

## **Database Confirmation**

**Description:** Confirm the database is valid and usable. Prepare documentation

**Acceptance Criteria:**

* Properly completed documentation (identified in Tasks)

**Tasks:**

* Add tasks to Trello/Jira
* Data Access Diagram
* Data Dictionary

## **Dashboard**

**Description:** Create a standardized dashboard that will be the basis upon which the entire system will operate

**Technology:** C# or Java Desktop App

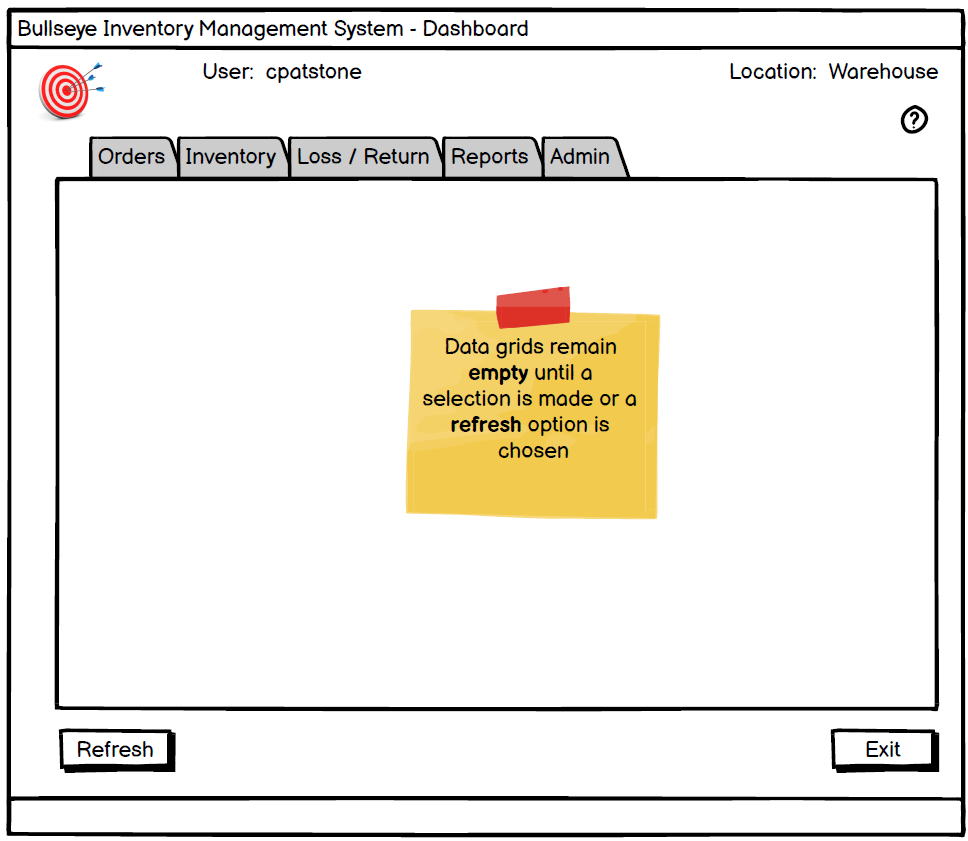
**Acceptance Criteria:**

* Proper UI practices, allows access to all functionality

**Type:** Desktop or Web App

**Tasks:**

* Add tasks to Trello/Jira
* Create dashboard wireframe
* Set up UX standards



## **Login**

**Actor(s):** All

**Description:** When any authorized user (any user with an active username and password) logs into the system

**Technology:** C# or Java Desktop App

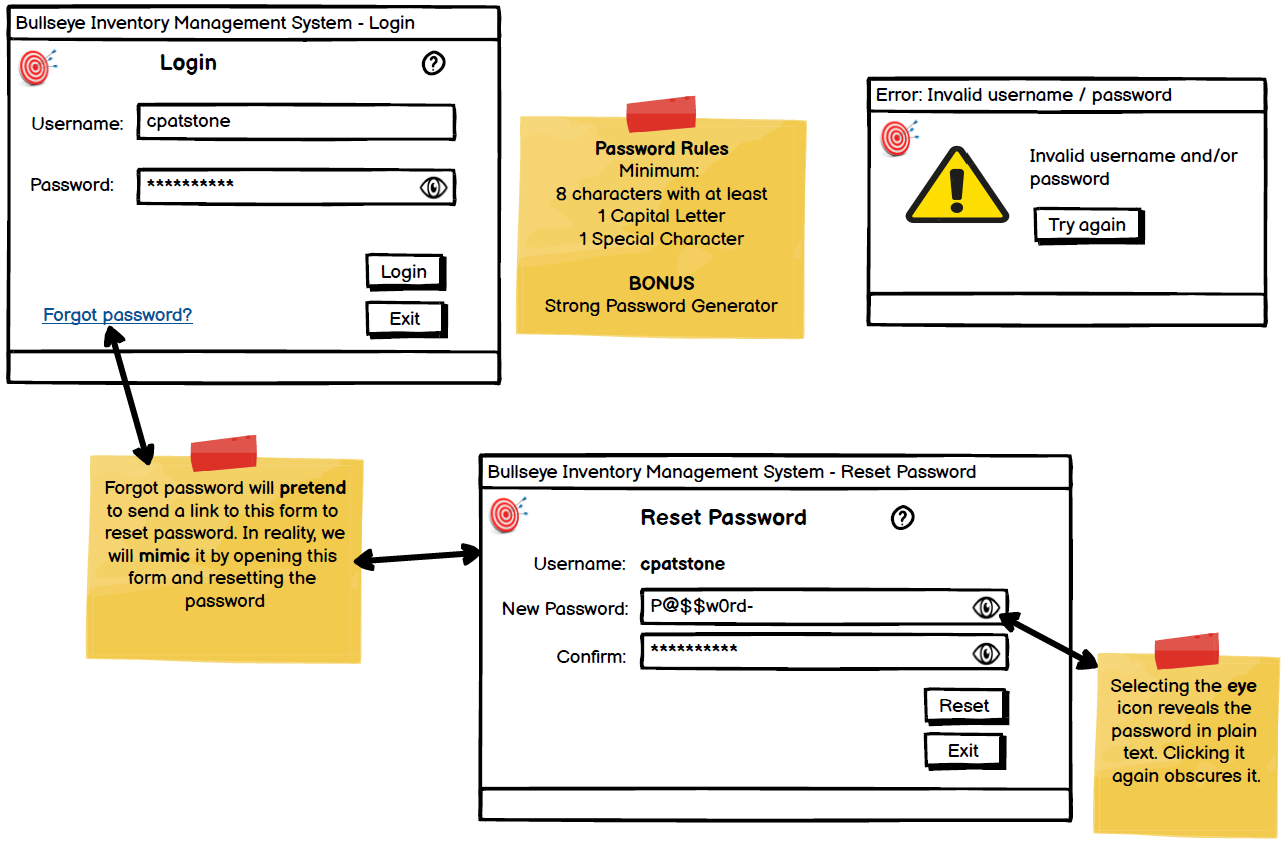
**Acceptance Criteria:**

* Every Bullseye employee must have a unique ID as well as a unique and valid Username and password to login to the system
* The unique ID shall be their first initial and last name (i.e. John Smith = jsmith), which shall also be their email address (jsmith@bullseye.com)
* After 3 incorrect attempts to login, set employee.locked = 1 (true) in DB
* INACTIVE users or users with locked = 1 should not be able to log in
* IF employee.active = 0 (false), user should receive a message saying “Invalid username and/or password. Please contact your Administrator admin@bullseye.ca for assistance”
* IF employee.locked = 1 (true), user should receive a message saying “You account has been locked because of too many incorrect login attempts. Please contact your Administrator at admin@bullseye.ca for assistance”

**Type:** Desktop or Web App

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Logout**

**Actor(s):** All

**Description:** When a user logs out of the system

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

* Manually: (click X in top right or form or use an ‘Exit’ button)
* Automatic: logout (after a period of time ~ 20 minutes?) - editable
* Any data or database connection cleanup process(es) need to be completed regardless of the method the user chooses to exit the application.

**Type:** Desktop or Web App

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Add User**

**Actor(s):** ADMIN

**Description:** Add a new unique user to the system

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

Username:

* Auto-generated - first initial and lastname (i.e. jperez)
* If a duplicate, add a 01 after (i.e. use jperez01 if jperez already exists)

Password:

* security requirements: minimum 8 characters, at least 1 non-numeric, one capital letter, one number
* must be encrypted in the database (i.e. not in plain text) - MD5 MINIMUM, 256 bit encryption preferred
* Default password (must be changed at first login: **P@ssw0rd-**)

ACTIVE= 1 by default

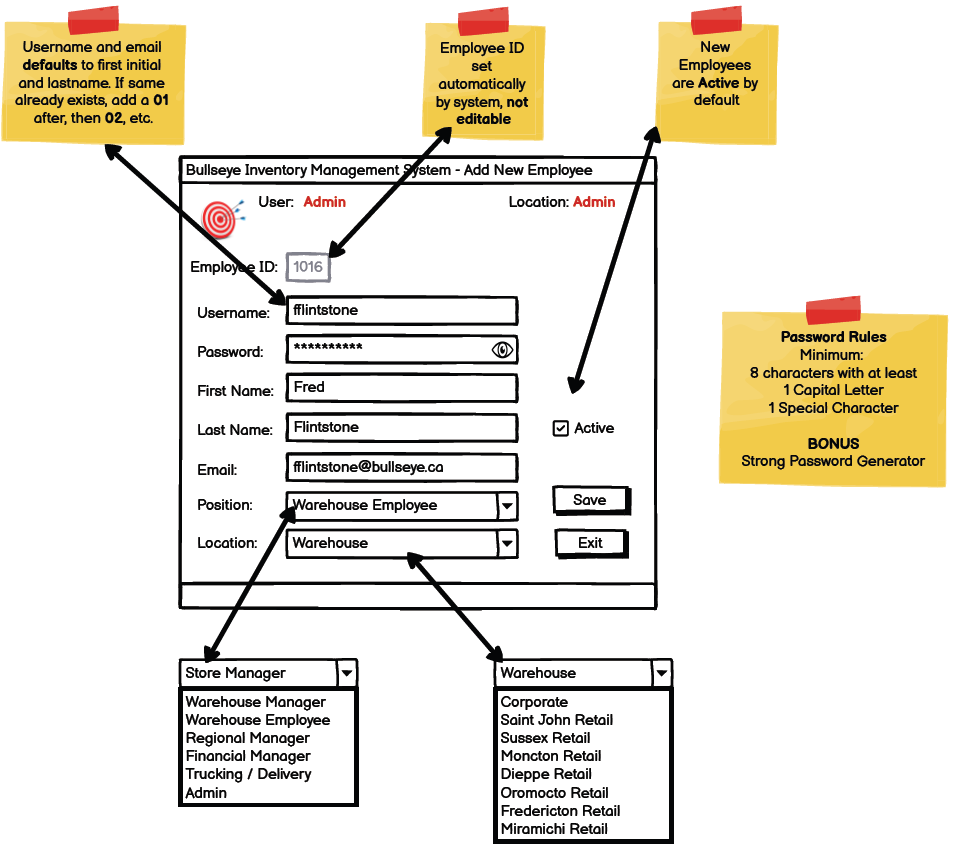
LOCKED: 0 by default

**Type:** Desktop App

**Permission(s):** ADMIN ONLY

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Edit User**

**Actor(s):** Admin

**Description:** Modify user info for system

**Technology:** C# or Java Desktop App

**Acceptance Criteria:** Changes take place in the employee table

Fields eligible for EDIT:

username \*\*\* MUST force following of username specs

Password \*\*\* MUST force following of password specs

FirstName

LastName

Email

active

positionID

siteID

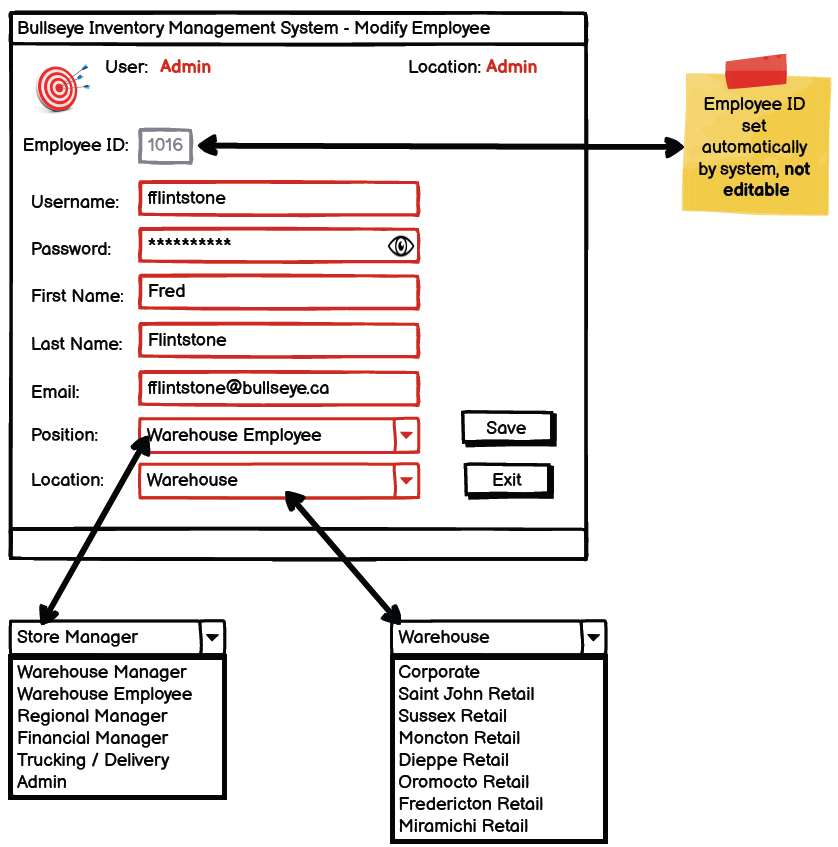
locked

**Type:** Desktop App

**Permission(s):** ADMIN ONLY

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Delete User**

**Actor(s):** Admin

**Description:** Remove a user from the system

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

User is never deleted, status set to INACTIVE.

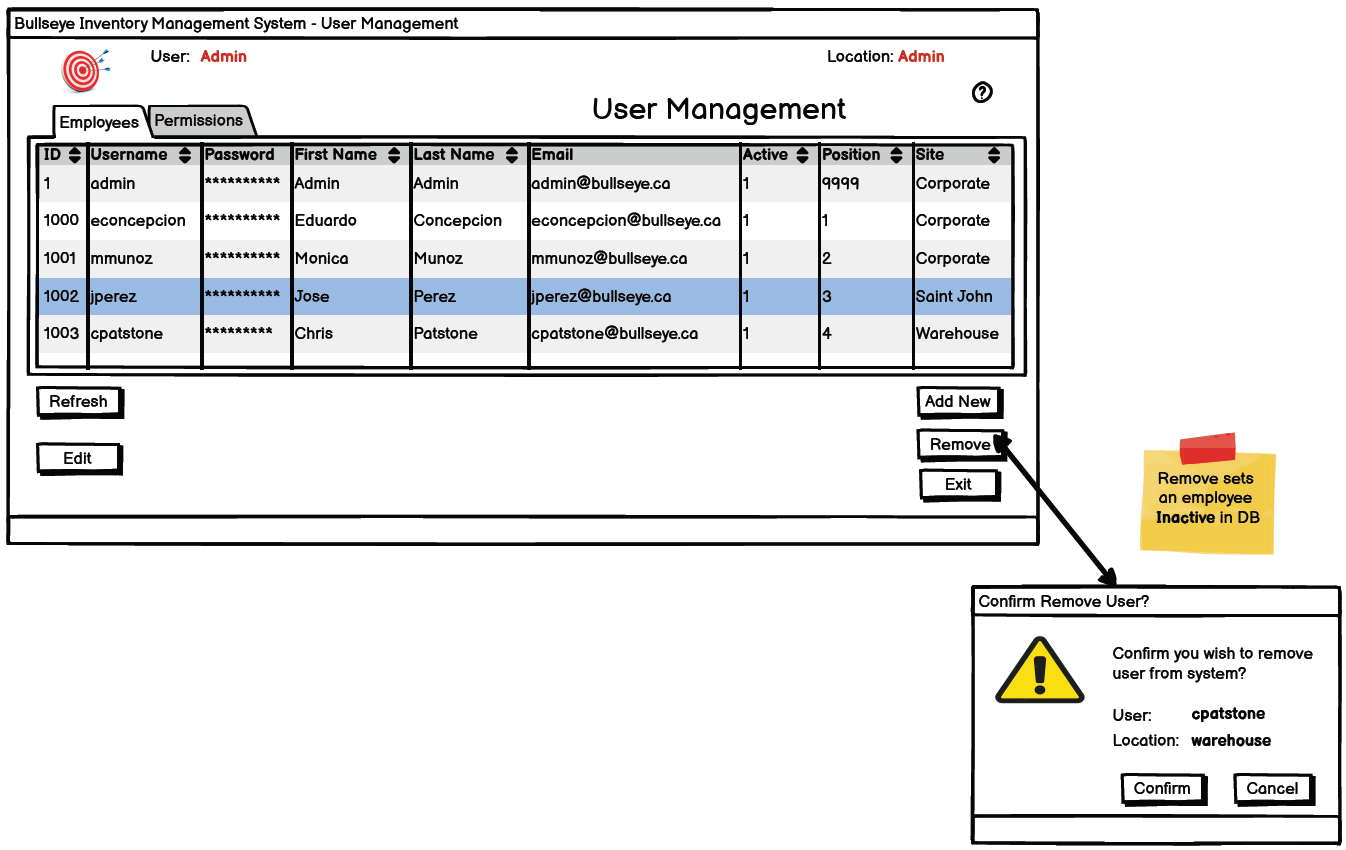
INACTIVE users should not be able to log in

**Type:** Desktop App

**Permission(s):** ADMIN ONLY

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Read User**

**Actor(s):** ALL

**Description:** Read (view) list and info about all employees

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

Username, location, role should be visible, password NOT visible.

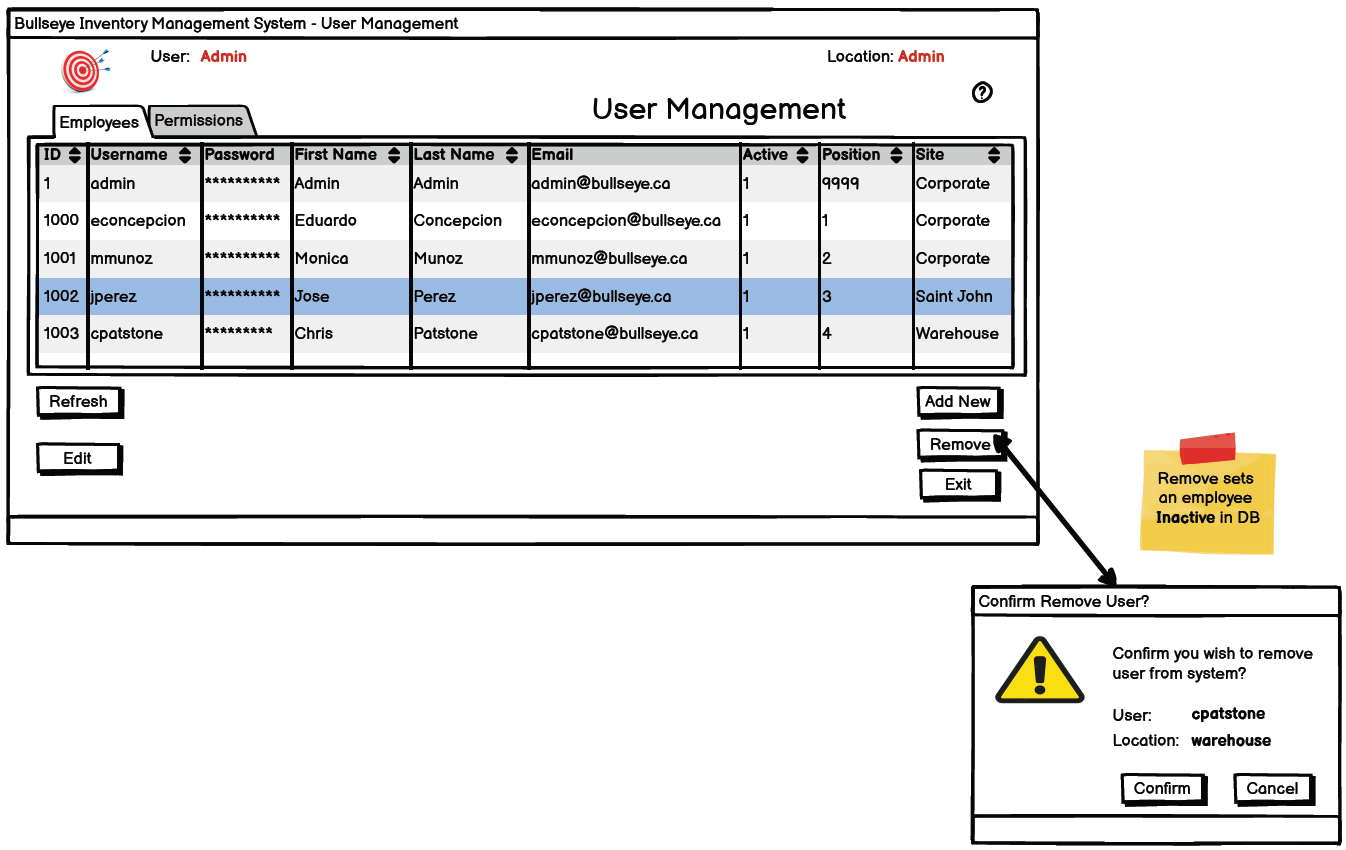
ALL users should have this permission (not customers, only authorized, active system users)

**Type:** Desktop or Web App

**Permission(s):** All

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Set User Permissions**

**Actor(s):** Admin

**Description:** Edit user permissions for any user in the system

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

ADMIN can assign a role to an employee.

**Type:** Desktop App

**Permission(s):** ADMIN ONLY

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Change Password**

**Actor(s):** ALL

**Description:** Allows any user to change their own password or force them to change their password

**Technology:** C# or Java Desktop App

**Acceptance Criteria:** default password is **P@ssw0rd-**

Password security requirements: minimum 8 characters, at least 1 non-numeric, one capital letter, one number

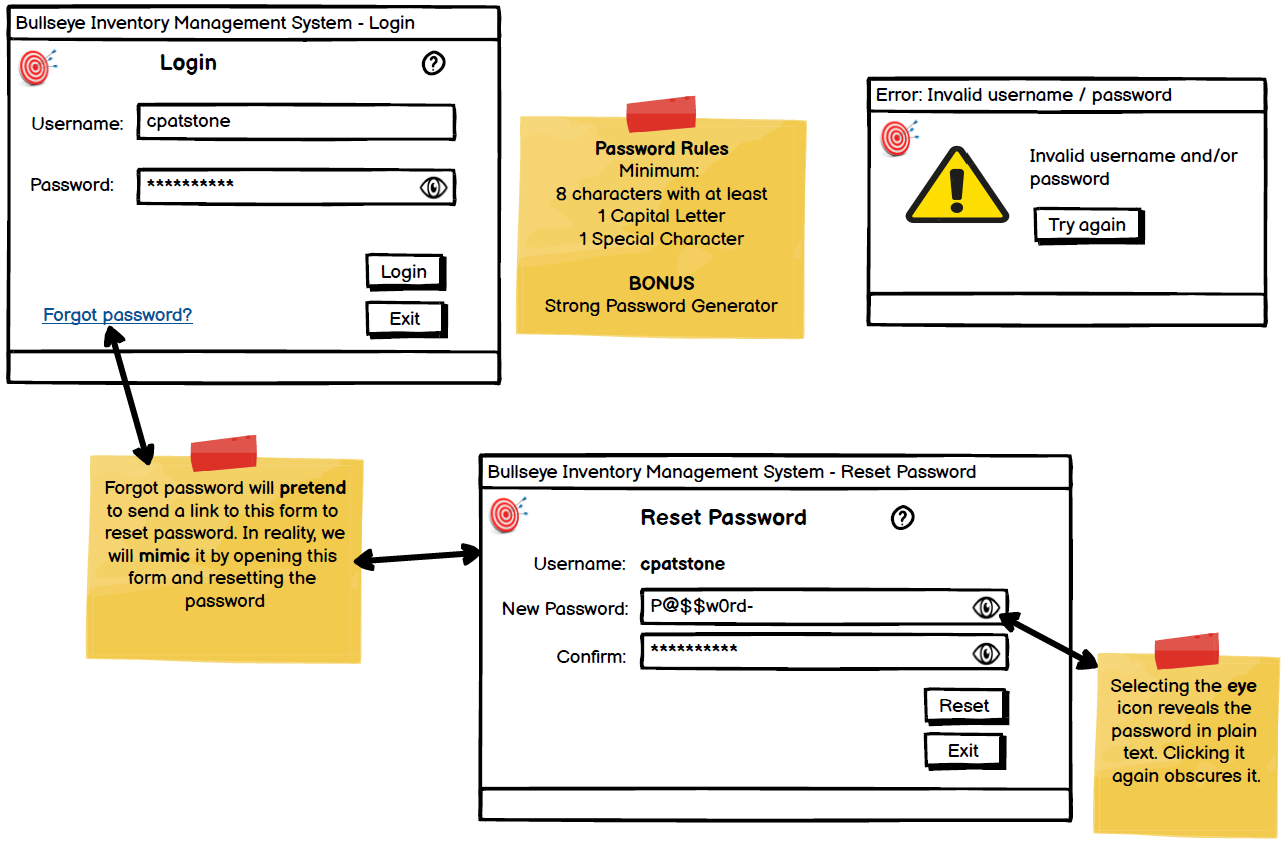
Password must be encrypted in the database (i.e. not in plain text) - MD5 MINIMUM, 256 bit encryption preferred. This is to prevent data breaches from having access to a list of passwords in plain text

**Type:** Desktop or Web App

**Permission(s):** None required, any **registered** ACTIVE user with a username and password can change their own password

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Edit Item**

**Actor(s):** Warehouse Manager

**Description:** Set any ‘Item’ to be inactive so that stores can no longer order that item (i.e. the item is no longer being produced, not available, etc.)

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

Ability to add image or description/notes field of item as well

**Type:** Desktop or Web App

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

# **SPRINT 2**

|  |  |
| --- | --- |
| Audit Activity  Move Inventory  Create Store Order  Create Emergency Order  Receive Store Order  Fulfil Store Order | Add Item to Backorder  View Store Order  Add Site  Edit Site  View Site  Edit Inventory |

## **Audit Activity**

**Actor(s):** System

**Description:** Every transaction that occurs in the system MUST have a corresponding audit record

**Technology:** Code-based and/or DB-based

**Acceptance Criteria:**

* Every action taken in the txn table creates a record of the activity in the Audit table.
* Info included in each audit record:
  + Unique Transaction ID,
  + userID who performed the task,
  + date/time,
  + description of the activity
* Can be set up as a generic trigger (minimum) or coded to handle all types of transactions with more specific descriptions (preferred)

**Type:** System, no GUI

**Permission(s):** N/A – Only System can perform this use case

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create code or database trigger to handle

## **Create Store Order**

**Actor(s):** Store Manager, Asst Store Manager, Warehouse Manager

**Description:** Create a new weekly order for a store (site).

**Technology:** C# or Java Desktop App OR Web-based

**Acceptance Criteria:**

* Order should be pre-populated with all items at or below the minimumReorderThreshold with an amount that brings them up to the optimumThreshold
* A new order can be created ONLY IF no order is currently open and active
* Only 1 NEW order of type ORDER or EMERGENCY can be active for each site (unlimited number in other statuses)
* When time to submit existing store order, should be options for both automatic (time-based) and manual option to submit

Process:

* Store Order items should be ***automatically*** added based on reorderThreshold. Any item quantity currently below the reorderThreshold for this site should be added automatically – by case size - enough to put it up to the ***optimumThreshold***
* Store Manager has option to edit quantity of any item(s) being ordered before submitting, add any other items, remove any items, etc.)

**siteIDTo** (where the order will be delivered)

**siteIDFrom** (***should default to warehouse***. Included in case of store-to-store transfers, but that feature has not been implemented)

**shipDate** (should default to be the ***next regular delivery date*** for the ***siteIDTo*** location)

**quantity** Quantities must be incremented/decremented for items by the ***listed case size***

**txnType:** ORDER

**txnStatus:** NEW or SUBMITTED

NEW: Store Manager has created a new store order, a database record is created

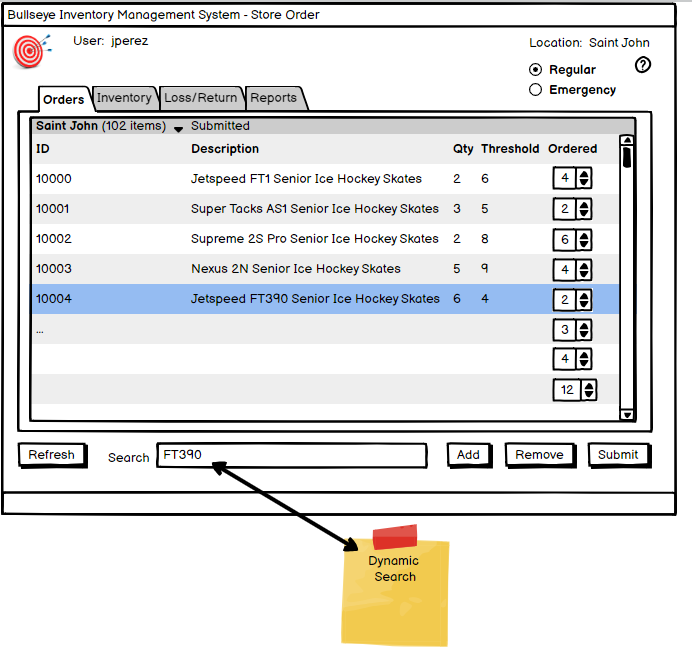
SUBMITTED: Store Manager has submitted the store order to the warehouse

**Type:** Desktop or Web App

**Permission(s):** Store Manager, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe (if appropriate)

****

## **Create Emergency Order**

**Actor(s):** Store Manager, Asst Store Manager, Warehouse Manager

**Description:** Create a new emergency order for a store (site).

**Technology:** C# or Java Desktop App OR Web-based

**Acceptance Criteria:**

* A new order can be created ONLY IF no existing emergency order is currently open and active
* Only 1 NEW order of type ORDER or EMERGENCY can be active for each site (unlimited number in other statuses)
* When time to submit existing store order, should be options for both automatic (time-based) and manual option to submit
* Maximum of 5 line items on an emergency order

Process:

* Store Manager has option to edit quantity of any item(s) being ordered before submitting, add any other items, remove any items, etc.)

siteIDTo (where the order will be delivered)

siteIDFrom (should default to warehouse. Included in case of store-to-store transfers, but that feature has not been implemented yet)

shipDate (should default to be the next regular delivery date for the siteIDTo location, then may be updated to be a different shipDate)

quantity Quantities must be incremented/decremented for items by the listed case size

**txnType:** EMERGENCY

**txnStatus:** NEW or SUBMITTED

NEW: Store Manager has created a new store order

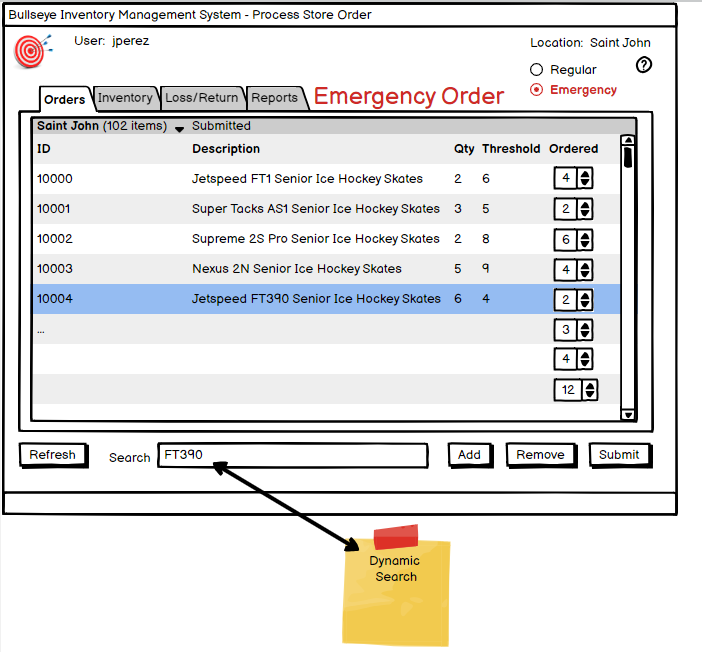
SUBMITTED: Store Manager has submitted the emergency order to the warehouse

**Type:** Desktop or Web App

**Permission(s):** Store Manager, Asst Store Manager, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Receive Store Order**

**Actor(s):** Warehouse Manager

**Description:** Warehouse Manager receives an order from store site

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

Obvious flag or notification showing there is a new order (txnStatus = SUBMITTED)

Warehouse Manager should by default see all OPEN orders (anything not COMPLETE or REJECTED)

Warehouse Manager clicks “Receive” to receive the order. txnStatus = RECEIVED

Warehouse Manager can review and assign different quantities or remove items that were ordered by the store based on availability, etc.

If the warehouse doe not have enough stock on hand to fulfil a store order, the missing item(s) and quantities are automatically added to a backorder that is assigned to that store when the Warehouse Manager receives the order

Once the Warehouse Manager has completed these tasks, they then click a button to send the txn to the warehouse floor for fulfillment (txnStatus = ASSEMBLING)

**txnType**: ORDER, EMERGENCY

**txnStatus**: SUBMITTED -> ASSEMBLING

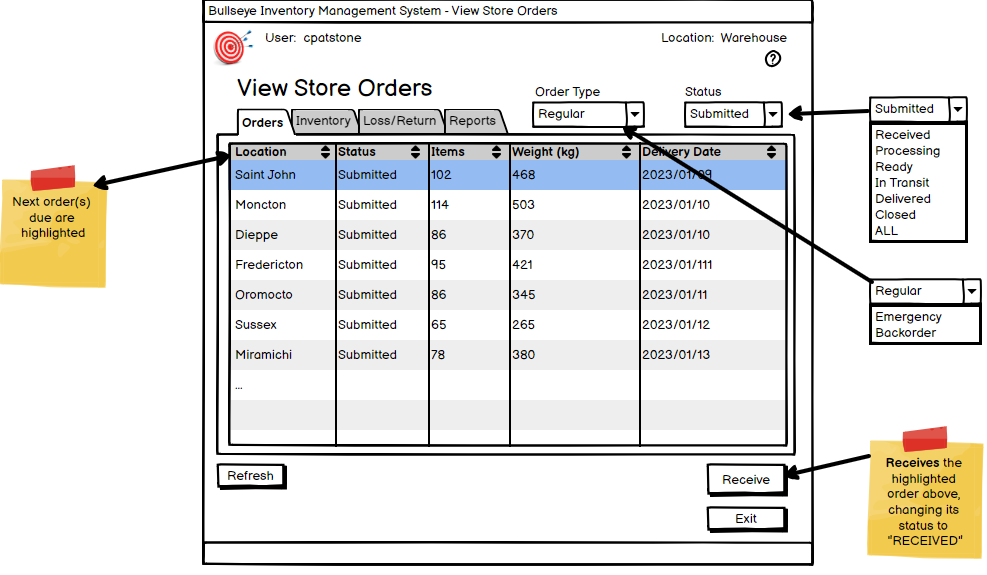
* SUBMITTED after order has been submitted by site and before it has been sent to warehouse workers for assembly
* ASSEMBLING while order is being prepared by warehouse staff

**Type:** Desktop or Web App

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Fulfil Store Order**

**Actor(s):** Warehouse Manager, Warehouse Worker

**Description:** Complete order for a store (site).

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

* This is the process of warehouse workers manually checking off items as they prepare the order.
* The manual checking off can be paper-based (a report) or on-screen, but ***each line item*** on the txn needs to be selected and quantity confirmed
* Once all items have been confirmed as picked, the tsnStatus is updated to txnStatus = ASSEMBLED

**txntype**: ORDER, EMERGENCY

**txnStatus**:

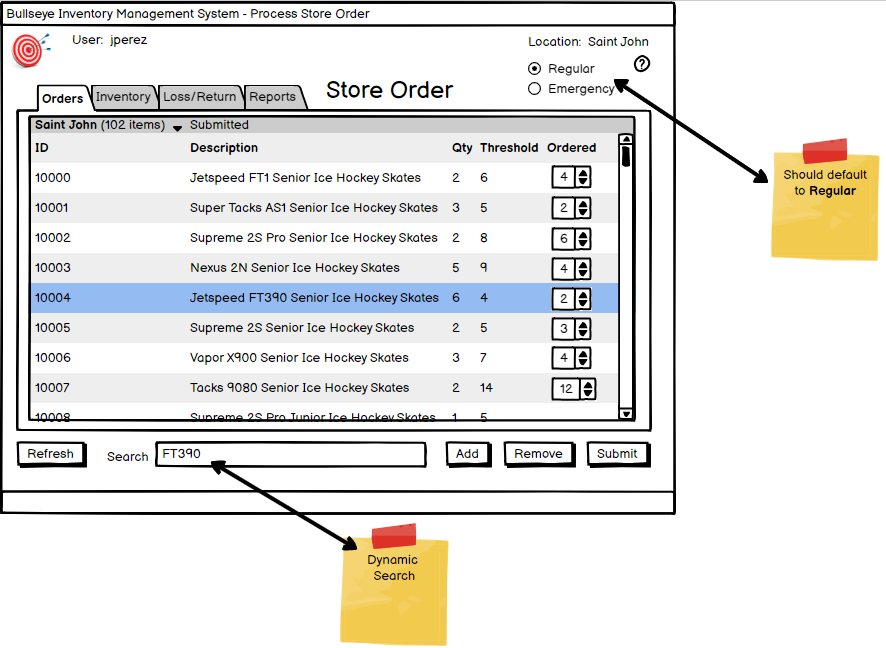
* ASSEMBLING while order is being prepared by warehouse workers
* ASSEMBLED once order is assembled at warehouse and ready for pickup

**Permission(s):** Warehouse Manager, Warehouse Worker

**Type:** Desktop App

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Move Inventory**

**Actor(s):** All

**Description:** Used whenever inventory changes “locations.” This is important to keep the inventory quantities accurate in the system.

**Technology:** Code-based solution

**Acceptance Criteria:**

* Change location of inventory (i.e. warehouse-to-order, order to truck, truck-to-store, inside a store, etc.)

Examples:

* Inventory is placed into an order as part of ‘Fulfil Store Order’ user story. That inventory is then removed from the Warehouse inventory (siteID = 2) and placed in the Warehouse Bay (siteID =3).
* Update the database as follows: siteIDTo=<ordering\_site>, siteIDFrom=<warehouse>, itemLocation = <orderid>,
* Inventory is loaded on a truck for delivery (removed from warehouse inventory onto truck inventory, siteID=<truck>, itemLocation = <orderid>)
* Inventory is received by a store (siteID=<ordering\_site>, itemLocation = “STOREROOM”)
* Inventory is moved within a store/warehouse (example: from a storeroom or shelf/aisle to another specific shelf or aisle, siteID=<site>, itemLocation = <new shelf in store>)

Included as part of the following Use Cases: Prepare Store Order, Fulfil Store Order, Pickup Store Order, Transport Store Order, Deliver Store Order, Accept Store Order

**Type:** System, no GUI

**Permission(s):** System

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe (if appropriate)

## **Add Item to Backorder**

**Actor(s):** Warehouse Manager, System

**Description:** Add an ordered item to a backorder. If a store orders a quantity of items that the warehouse cannot provide, a backorder is automatically created for those items. The warehouse manager can also create a backorder manually.

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

* If no backorder exists, create when first item added for a site
* If an existing backorder already exists for this store, add items to that backorder instead of creating a new backorder
* Default shipDate is the NEXT standard delivery day for that site
* shipDate can also be modified manually by Warehouse Manager
* quantity must be incremented/decremented by case size

**txntype** = BACKORDER

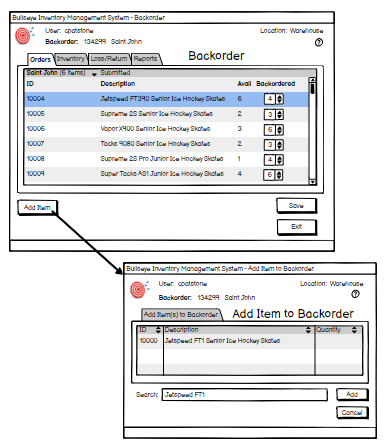
**txnStatus** = New

**Type:** Desktop or Web App

**Permission(s):** Warehouse Manager, System

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **View Store Order**

**Actor(s):** Warehouse Manager, Store Manager

**Description:** View all outstanding store orders

**Technology:** C# or Java Desktop App for Warehouse, C# or Java-based OR Web-based for Stores

**Acceptance Criteria:**

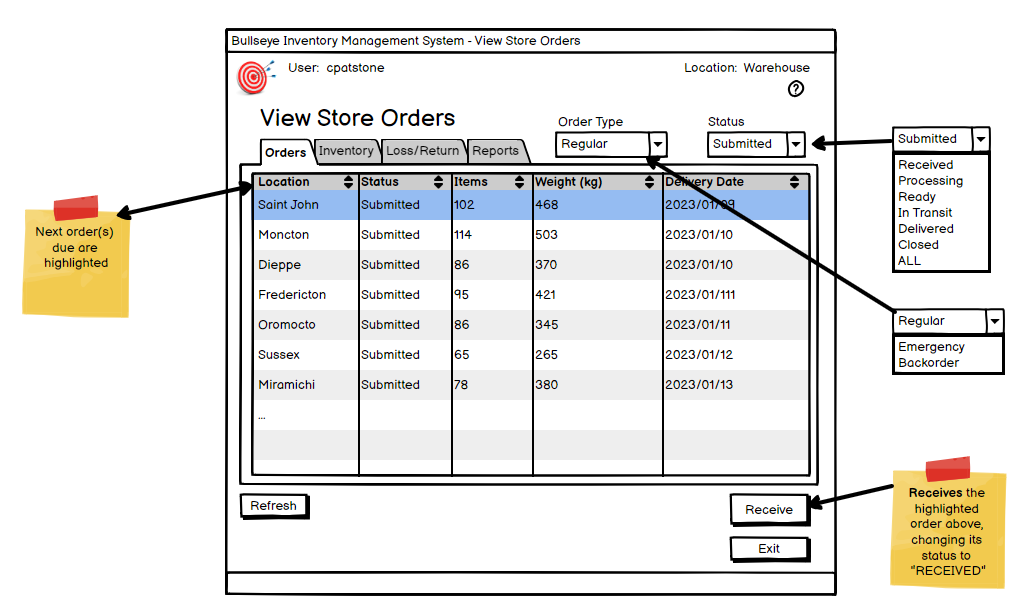
* Default show open store orders, can toggle all or a specific status
* Default to show orders from user’s site, can toggle all or a specific site
* Warehouse Manager should be able to see ALL orders, default to open
* Default view should be to show only orders with a txnStatus active order (i.e. txnStatus NOT CLOSED or CANCELLED), but with capability to filter to include closed or cancelled orders

**Type:** Desktop or Web App

**Permission(s):** Warehouse Manager, Store Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **View Site**

**Actor(s):** ALL

**Description:** View all company sites (stores, warehouses, corporate offices, etc.)

**Technology:** C# or Java Desktop App for Warehouse, C# or Java-based OR Web-based for Stores

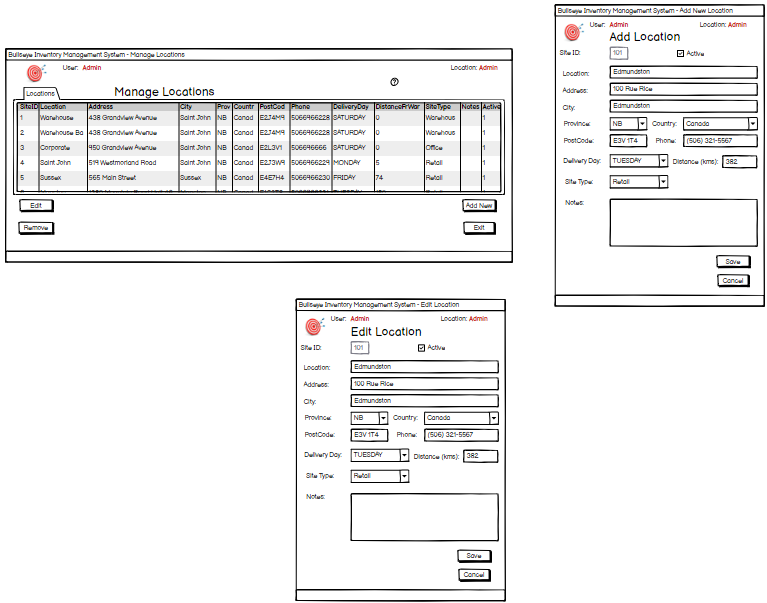
**Acceptance Criteria:** All Users (including any customers online): Read only

**Type:** Desktop or Web App

**Permission(s):** Any logged-in Bullseye user (NOT Acadia/external)

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Add Site**

**Actor(s):** Admin

**Description:** Add a new site (store, warehouse, corporate office, etc.)

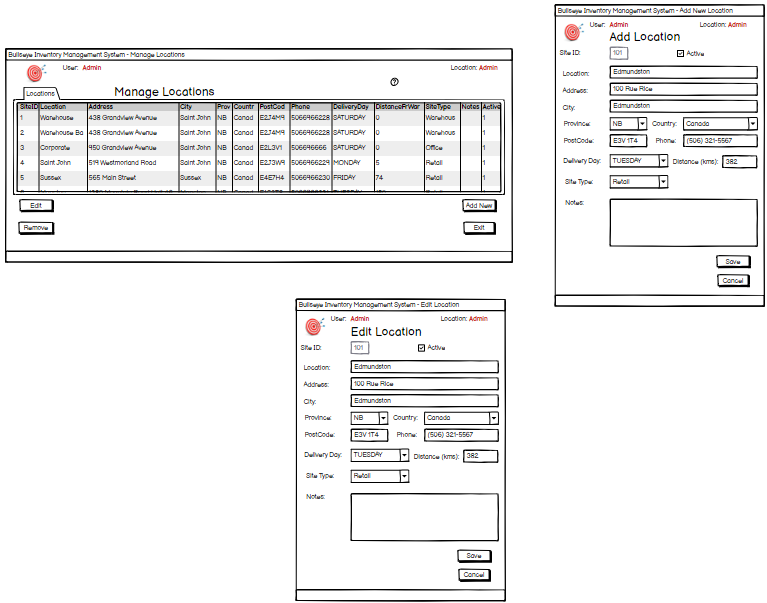
**Acceptance Criteria:** Admin can add new sites as the company grows. site.active = 1 is default for new sites.

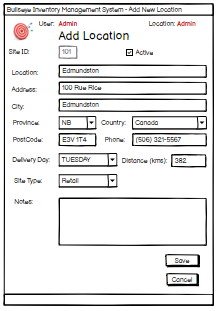
**Technology:** C# or Java Desktop App

**Permission(s):** Admin

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



****

## **Edit Site**

**Actor(s):** Admin

**Description:** Edit info about an existing site (store, warehouse, corporate office, etc.)

**Technology:** C# or Java Desktop App

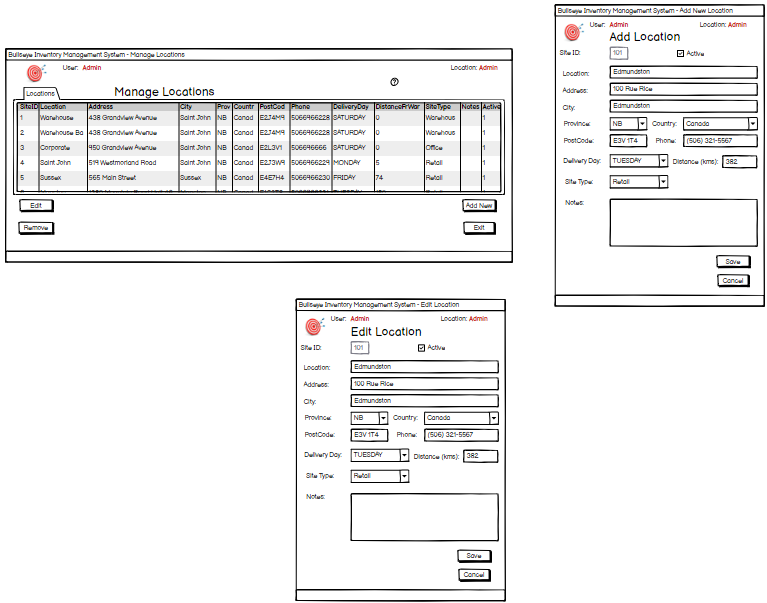
**Acceptance Criteria:**

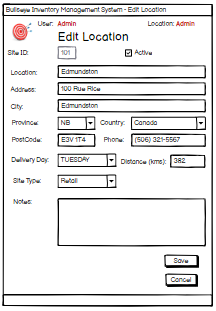
**Type:** Desktop App

**Permission(s):** Admin

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe





## **Edit Inventory**

**Actor(s):** Store Manager, Warehouse Manager

**Description:** STORE MANAGER or WAREHOUSE MANAGER can modify ONLY the reorderThreshold and optimumThreshold for inventory at their particular site

**Technology:** C# or Java Desktop App for Warehouse, C# or Java-based OR Web-based for Store

**Acceptance Criteria:** Store Managers can only change the reorderThreshold and optimumThreshold for their own store - i.e. Store Manager of Moncton retail store can modify the reorderThreshold for any inventory item in the Moncton store only.

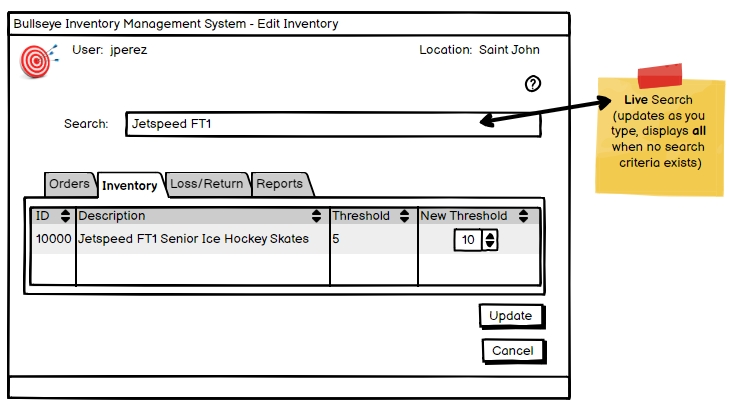
Ability to add/edit notes of inventory item as well

**Type:** Desktop or Web App

**Permission(s):** Store Manager, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



# **SPRINT 3**

|  |  |
| --- | --- |
| Check Delivery  Pickup Store Order  Transport Store Order  Deliver Store order  Accept Store Order  Edit Supplier  Add Supplier | Place Online Order  View Online Order  Prepare Online Order  Receive Online order  Add Supplier  Modify Record |

## **Check Delivery**

**Actor(s):** Acadia

**Description:** Acadia checks a **web portal** to see what orders are going out for delivery on a specific day or week

**Technology:** Web-based

**Acceptance Criteria:**

Report shows what order(s) go out each day.

Deliveries NEED to account for ***weight*** of each order to identify TRUCK size for ALL orders going out on a particular day.

**Type:** Web App

**Permission(s):** Acadia

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Pickup Store Order**

**Actor(s):** Driver, Warehouse Worker

**Description:** Driver Picks up order(s). Normally pickups are at warehouse and taken to a store (site).

**Technology:** C# or Java Desktop App for Warehouse, C# or Java-based OR Web-based for Store

**Acceptance Criteria:** Items transferred from warehouse inventory loading bay to vehicle inventory (siteID=11 -> siteID=1, *itemLocation=”ON TRUCK”*)

**txnTypes**: ORDER, EMERGENCY

**txnStatus**: ASSEMBLED, IN TRANSIT

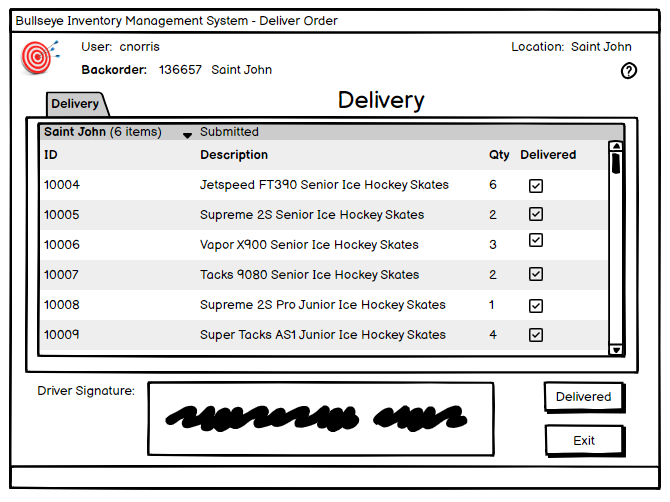
* ASSEMBLED once order is assembled at warehouse and ready for pickup
* IN TRANSIT while order is on truck and on way to store for delivery

**Type:** Desktop App

**Permission(s):** Driver, Warehouse Worker

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Transport Store Order**

**Actor(s):** Driver

**Description:** Transport order to a store site

**Technology:** Code-based

**Acceptance Criteria:** This is the actual transport phase of the order being taken from one site to another (normally warehouse to store)

**txnTypes**: ORDER, EMERGENCY

**txnStatus**: IN TRANSIT

**Type:** System, manual process, no GUI

**Permission(s):** Driver

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe (if appropriate)

## **Deliver Store Order**

**Actor(s):** Driver, Store Worker, Store Manager

**Description:** Deliver order to a store site

**Technology:** C# or Java Desktop App for Warehouse, C# or Java-based OR Web-based for Store

**Acceptance Criteria:** This is the actual “delivery” portion of the order arriving at the store and being offloaded. It involves the Driver and store staff. Items transferred from truck inventory to store inventory (siteID=1 -> siteID=<storeid>)

**txnTypes**: ORDER, EMERGENCY

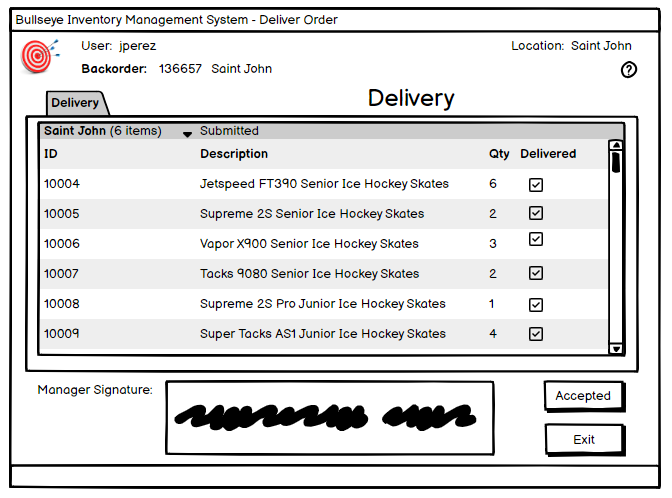
**txnStatus**: DELIVERED

**Type:** Desktop App

**Permission(s):** Driver, Store Worker, Store Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe (if appropriate)



## **Accept Store Order**

**Actor(s):** Store Worker, Store Manager, Assistant Store Manager

**Description:** Store accepts an order from a delivery driver or a courier

**Technology:** C# or Java Desktop App OR Web-based

**Acceptance Criteria:**

Items added to the order should be removed from the truck inventory and added to the site inventory (siteID=<ordering\_siteId>, itemLocation=” STOREROOM”

**txnType**: ORDER, EMERGENCY

**txnStatus**:

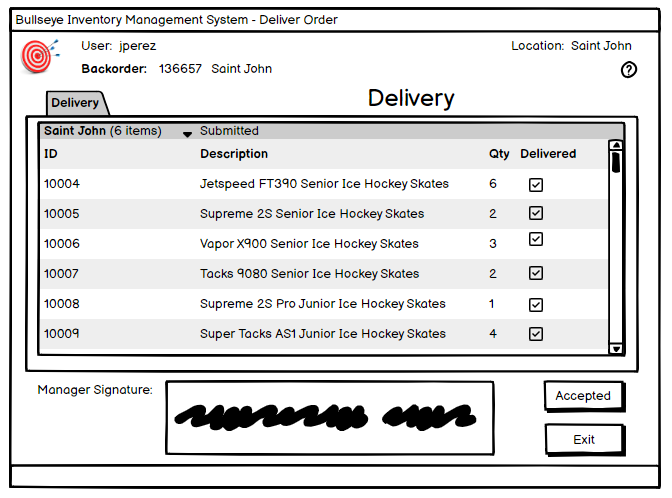
* DELIVERED after order has been delivered to the site and delivered by the delivery truck
* CLOSED after order has been received by store and accepted into inventory

**Type:** Desktop App

**Permission(s):** Store Manager, Assistant Store Manager, Store Worker

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Place Online Order**

**Actor(s):** Customer

**Description:** Any customer can visit the Bullseye online storefront and order any items that are in stock at the selected location for curbside pickup.

**Technology:** Web-based

**Acceptance Criteria:**

Customer mustselect a store to place their online order with so they can see if sufficient inventory available.

That will display the current inventory levels for that store while the customer is searching so that they can see if a product is not available / not in stock

Ability to bring up an item page that also shows an image of the product

Ability to add item(s) to a cart so that multiple item(s) can be ordered at once

Customers submit:

name, email address, and phone number.

(Just add a couple of text boxes to the web page that captures that info and add that info to the Notes field in the transaction)

**txnType** = ONLINEORDER

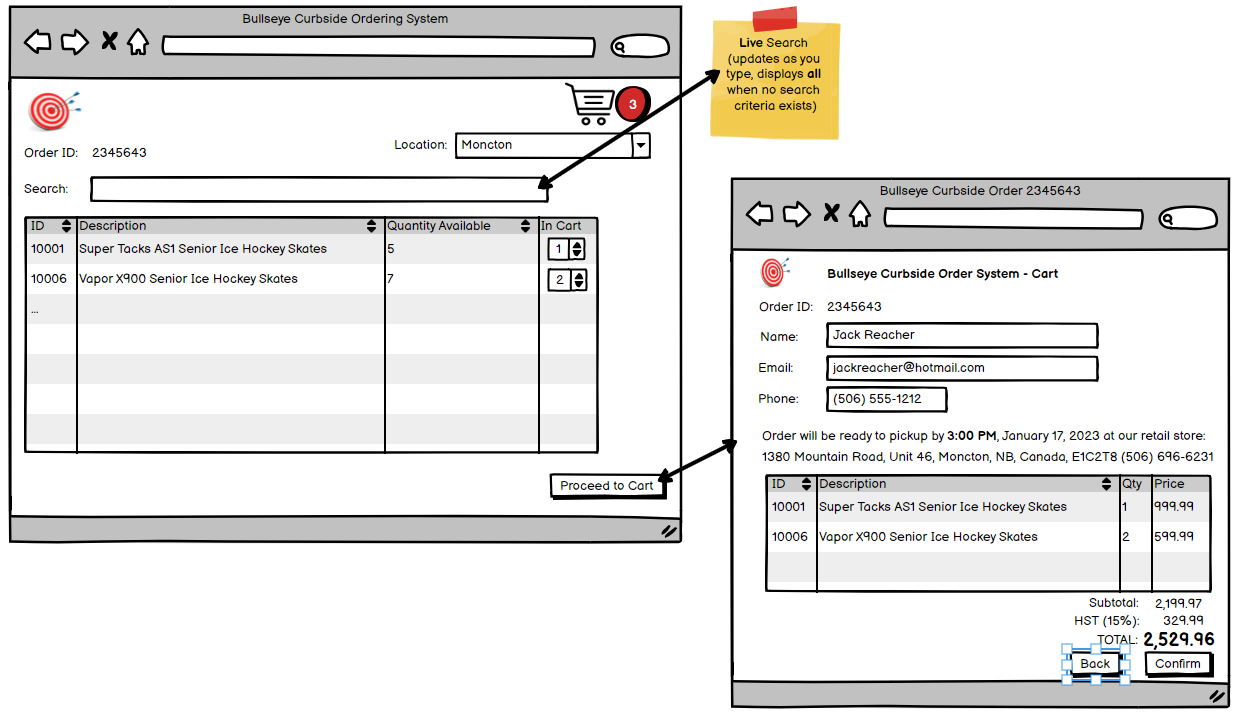
**vehicleType** = Pickup

**Type:** Web App

**Permission(s):** None required

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **View Online Order**

**Actor(s):** Customer

**Description:** Any customer can search for the current status (txnStatus) of their order by entering the txnID or their email address into a user-friendly online search

**Acceptance Criteria:** Allow search by email address (stored in the txn Notes)

**txnType:** ONLINEORDER

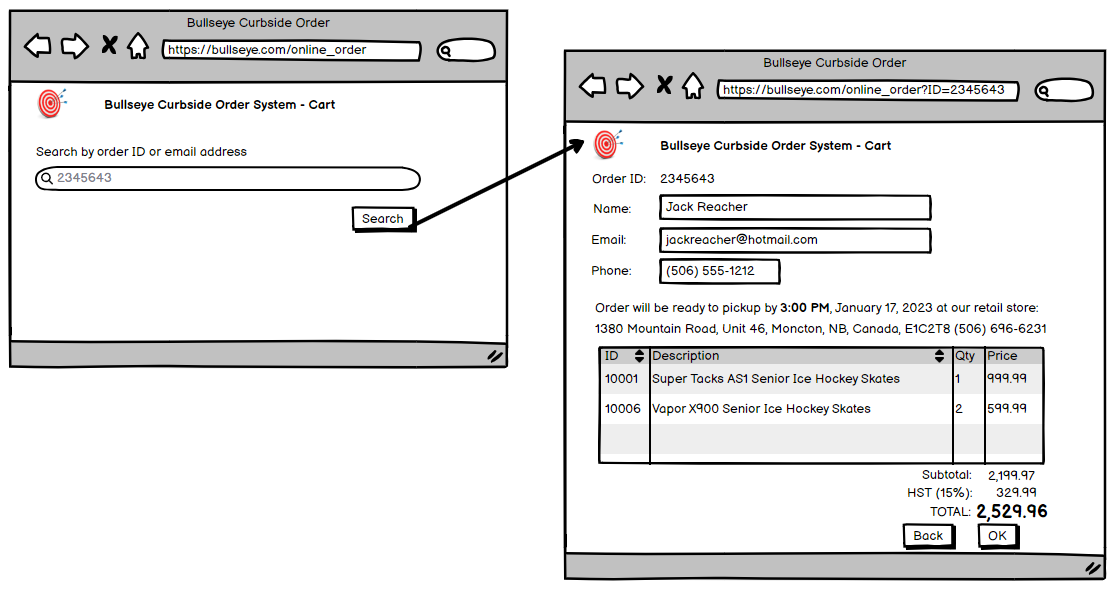
**Technology:** Web-based

**Type:** Web App

**Permission(s):** None required

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Prepare Online Order**

**Actor(s):** Store Manager, Store Worker, Customer

**Description:** Store receives and prepares to fulfil an online order

**Technology:** C# or Java Desktop App OR Web-based for Store

**Acceptance Criteria:**

* Any Store Manager or Assistant store manager can see new online orders for their store (site)
* Any Store Manager or Assistant store manager can fulfil new online orders for their store (site)
* Orders should be fulfilled similarly to how warehouse workers fulfil a store order - make the employer check off each item as it is gathered for the order until it is complete and Ready
* Use same statuses as for store orders, except no requirement for ‘IN TRANSIT’ status

**txnType:** ONLINEORDER

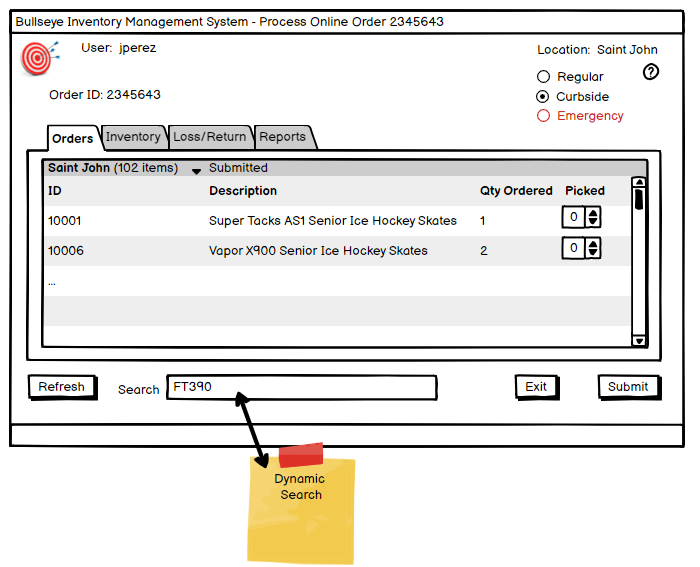
**vehicleType** = Pickup

**Type:** Desktop App

**Permission(s):** Store Manager, Store Worker

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Receive Online Order**

**Actor(s):** Store Manager, Store Worker, Customer

**Description:** Curbside pickup at appropriate store when status = READY

**Technology:** C# or Java Desktop App OR Web-based for Store

**Acceptance Criteria:** Customer can enter their order id (txn id) online and see the current status of the order.

**txnstatus**: READY -> CLOSED once delivered curbside

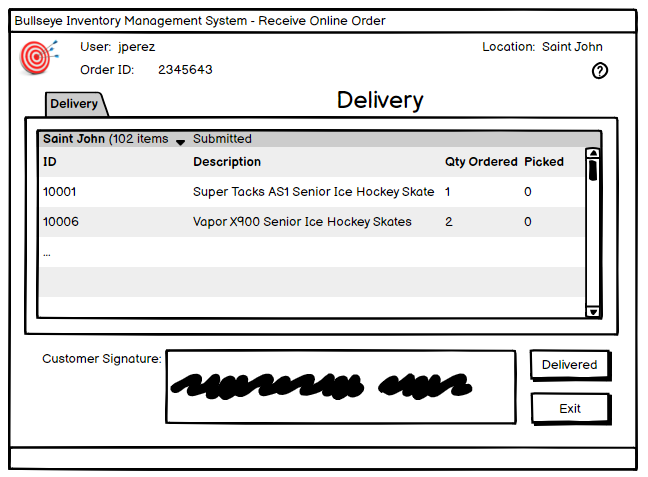
* READY once order is assembled at warehouse and ready for pickup
* CLOSED order has been picked up by customer

**Type:** Desktop or Web App

**Permission(s):** None required

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Modify Record**

**Actor(s):** Admin

**Description:** Allows an ADMIN to modify data - ONLY in the **txn** table

**Technology:** C# or Java Desktop App

**Technology:** C# or Java Desktop App

**Acceptance Criteria:** This feature is ONLY FOR non-closed orders/transactions

All data in the txn record can be seen

Only these fields may be modified:

siteIDTo, siteIDFrom, txnStatus, shipDate, txnType, barCode, deliveryID, emergencyDelivery,

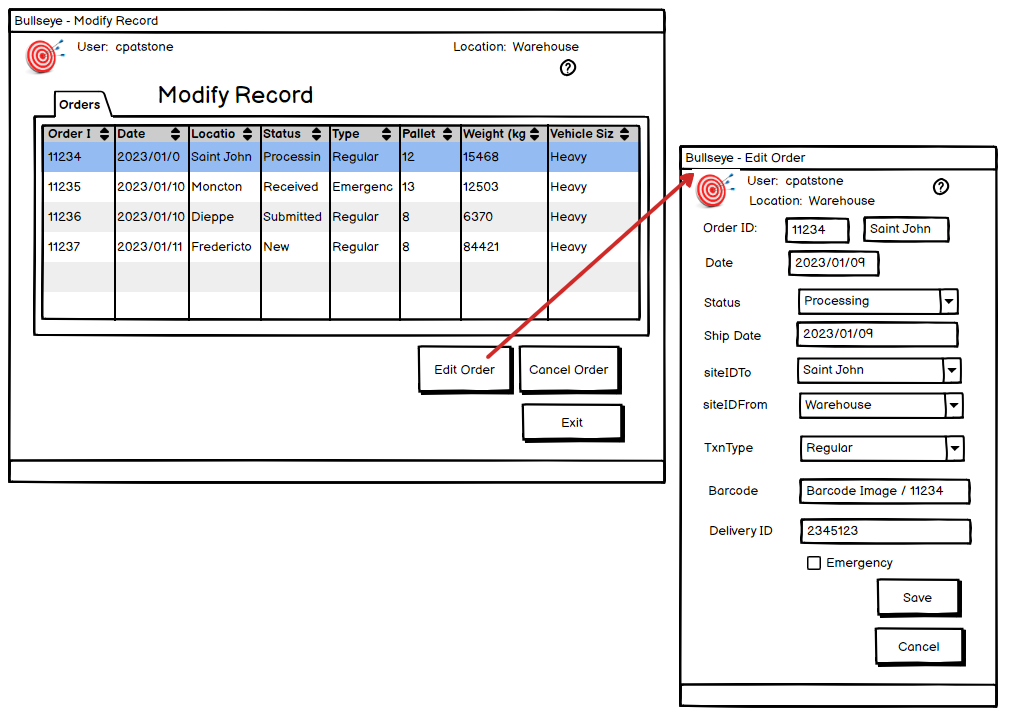
**txnStatus:** CANCELLED if you want to ‘delete’ a txn (You can never actually delete), otherwise none

**Type:** Desktop App

**Permission(s):** ADMIN ONLY

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



## **Add Supplier**

**Actor(s):** Warehouse Manager

**Description:** Admin or Warehouse Manager can add a new supplier to the list of possible suppliers

**Technology:** C# or Java Desktop App

**Acceptance Criteria:** supplier.active = 1 (ACTIVE by default). Any supplier with active = 0 cannot be seen/ordered from

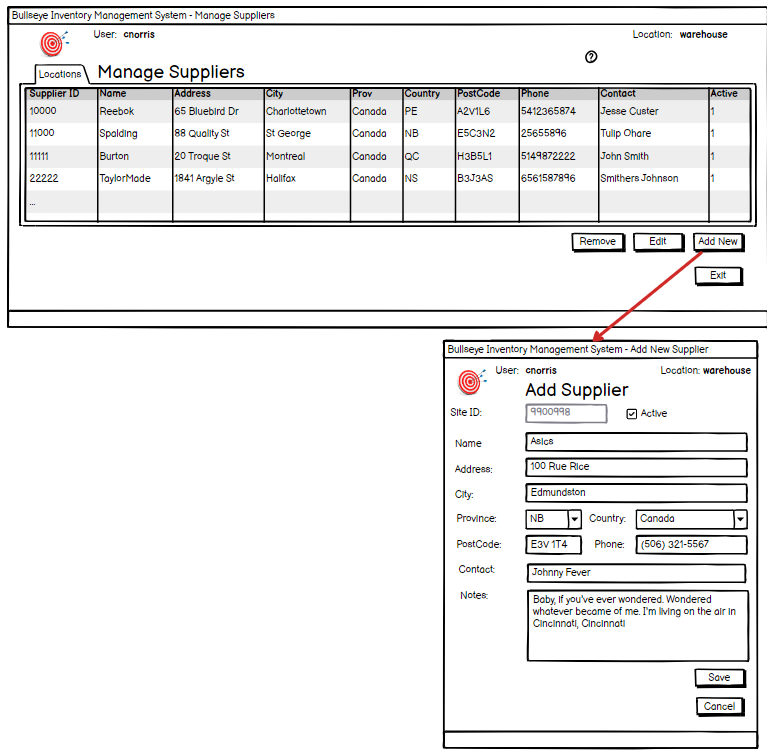
Ability to add/edit notes about supplier

**Type:** Desktop App

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

****

## **Edit Supplier**

**Actor(s):** Warehouse Manager

**Description:** Admin or Warehouse Manager can edit any existing supplier

**Technology:** C# or Java Desktop App

**Acceptance Criteria:** supplier.active = 1 (ACTIVE by default). Any supplier with active = 0 cannot be seen/ordered from

Ability to change information in supplier table

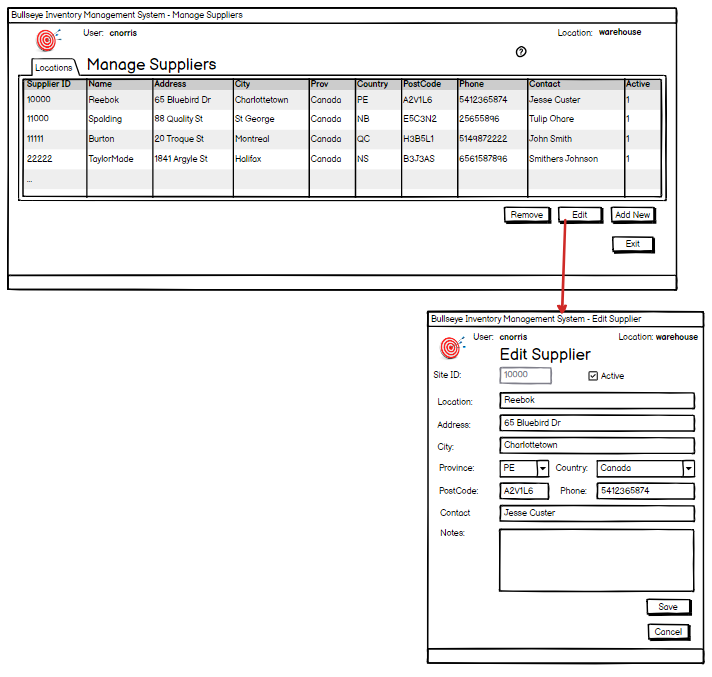
Ability to add/edit notes about supplier

**Type:** Desktop App

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe



# **SPRINT 4**

## **Create Loss**

**Actor(s):** Store Manager, Warehouse Manager

**Description:** Create a record to track lost inventory

* LOSS includes stolen, missing
* DAMAGE includes damaged inventory
* Create an explanatory note

**Technology:** C# or Java Desktop App OR Web-based for Store

**Acceptance Criteria:**

Item is successfully recorded as lost

Inventory is updated appropriately

Ability to add a note to explain the situation

**txntype**: LOSS or DAMAGE

**Type:** Desktop App

**Permission(s):** Store Manager, Asst Store Manager, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Process Return**

**Actor(s):** Store Manager, Warehouse Manager

**Description:** Create a record to track returned inventory item(s) (not orders, item(s))

**Technology:** C# or Java Desktop App OR Web-based for Store

**Acceptance Criteria:**

* This is when a customer returns something to the store
* Allows the store to add the item back to inventory for re-sale \*if\* in good condition
* Include an explanatory note
* Product is added back to inventory \*if\* in good condition
* Once item is recorded as returned, a LOSS may be created \*if\* the item(s) are not in good condition and cannot be re-sold
* Inventory is updated appropriately

**txnType**: RETURN

**Type:** Desktop App

**Permission(s):** Store Manager, Asst Store Manager, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Add New Product**

**Actor(s):** Warehouse Manager

**Description:** Add new supplier product to inventory list so that store managers can add them to their weekly orders and emergency orders.

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

* status: item.active = 1 (ACTIVE by default)
* If active = 0, the item can no longer be ordered
* Ability to add image of product

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Edit Product**

**Actor(s):** ADMIN, Warehouse Manager

**Description:** Edit inventory item information

**Technology:** C# or Java Desktop App OR Web-based for Store

**Acceptance Criteria:**

* Edit ONLY the following fields: name, description, category, supplierID, case size, weight, active, notes
* Delete = INACTIVE (can never remove, only set inactive)
* Ability to bring up a screen showing individual item info, including image
* Ability to add a different image of product

**Permission(s):** Admin, Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Create Supplier Order**

**Actor(s):** Warehouse Manager

**Description:** Warehouse Manager creates an order

**Technology:** C# or Java Desktop App

**Acceptance Criteria:**

* Inventory items automatically added when below minimum threshold (similar to store orders). Any item(s) can also be added manually
* Search capabilities required
* Ability to Sort/filter by supplier
* Ability to bring up a screen showing individual item info, including image

**Permission(s):** Warehouse Manager

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframe

## **Create Reports**

**Actor(s):** Manager, Warehouse Manager, Store Manager, System

**Description:** Create a dynamic report based on provided templates (see templates below)

**Technology:** PowerBI, MicroStrategies, etc.

**Acceptance Criteria:**

ALL reports should allow for dynamic date ranges (i.e. datetimepickers for start and end dates for a time frame) when applicable

1. **Delivery Report** - by day (selectable day of week), for each individual delivery, used as a record by Acadia and driver to see where they are going each day. INCLUDE routes/locations, mileage, weight, and vehicle size
2. **Store Order** - list report for each individual order, used as a record for store
3. **Shipping Receipt** - for each individual order, used as a record by driver
4. **Inventory** - sort by ALL or by individual store
5. **Orders** - sort by ALL or by individual store (Summary report, not each product detail)
6. **Emergency Orders** - sort by ALL or by individual store
7. **Users** - list all, sort by role, site
8. **Backorders** - sort by ALL or by individual store
9. **Supplier Order** (sort/filter by and/or page break between suppliers, used to send to the supplier to order more items)

Use an appropriate reporting tool (Tableau, Data Analysis tool, HTML, Crystal Reports, Java Reporting Tools, etc.). DOES NOT need to be accessible directly from within your system (i.e. you can open Tableau or one of the other tools to get to your reports)

Users: ALL with system access can view reports

**Type:** Desktop or Web App

**Permission(s):** Manager, Warehouse Manager, Store Manager, System

**Tasks:**

* Identify tasks required to complete this user story
* Add all tasks to Trello/Jira
* Create wireframes