Bullseye Case Study

PROG1197i Individual Software Development Project (ISDP)

Specifications Manual

Winter 2023

Chris London & Aaron Mitchell

Last Update: Mar 21, 2023

Contents

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, triggers, stored procedures, etc.,** to the database to handle any functionality you may wish to add, but **you may not alter 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
- Four (4) sprints with a series of features/requirements to be completed, as listed in this document
- **Assessment:**
  - Log/Journal 5%
  - Sprint 1 20%
  - Sprint 2 25%
  - Sprint 3 25%
  - Sprint 4 25%
- Formal Sprint Planning sessions will occur at the start of each sprint (attendance is mandatory)
- Weekly demonstrations of your progress will occur during scheduled class times
- Formal Sprint Reviews will occur at the end of each sprint (attendance is mandatory)
- Grades based on functionality based on given specifications. 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
- No grades are given for inefficient code, commenting, poor UX / design. At this point in your career, these are expected and while you will not receive credit for doing it, you will be penalized for not following specs, not writing efficient code, not commenting, and not following good UX / design standards
- Every sprint is REQUIRED to successfully complete this course (i.e. you cannot decide not to submit a sprint and be successful)
- 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 permission-based access. To access a particular aspect of the system, the user logged in must have the appropriate permission(s). Each feature/functionality will have the appropriate permission(s) listed with the other specifications. Most of these permissions are self-explanatory.

System Permissions:

ACCEPTSTOREORDER
ADDITEMTOBACKORDER
ADDNEWPRODUCT
ADDSITE
ADDUSER
CREATEBACKORDER
CREATELOSS
CREATEREPORT
CREATESTOREORDER
CREATESUPPLIERORDER
DELETELOCATION
DELETEUSER
DELIVERY
EDITINVENTORY
EDITITEM
EDITPRODUCT
EDITSITE
EDITUSER
FULFILSTOREORDER
MODIFYRECORD
MOVEINVENTORY
PREPARESTOREORDER
PROCESSRETURN
READUSER
RECEIVESTOREORDER
SETPERMISSION
VIEWORDERS

Coding Specifications

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 to make this a smooth experience.

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

Examples: postal code, phone number, email address

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, siteType, txnType, txnStatus, posn

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

Sprint 1

Sprint 1 is designed to set up the basics and lay the groundwork for the entire system. The following requirements are included in Sprint 1:

GitHub Project Repository

1. Create a new ***private*** repository in your GitHub account (if you do not have a GitHub account or cannot remember the one you created in Dev Tools, create a new one and use that).

2. Name your new repository as follows:
   a. **isdp2023_yourfirstnameyourlastname** (Example: **isdp2023_ChrisLondon**)

3. Invite your instructors to have access to your repository. You can do this inside your GitHub repository by going to Settings > Collaborators > Add by email

   Invite your instructors by using the following addresses:

   a. Chris.London@nbcc.ca
   b. Aaron.Mitchell@nbcc.ca

4. This 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.

Data Access Diagram

Create a Data Access Diagram (ER Diagram) using the existing Database Schema. You can obtain this via the MySQL Workbench. Example of how to do this can be found here:

https://medium.com/@tushar0618/how-to-create-er-diagram-of-a-database-in-mysql-workbench-209fbf63fd03

Remember to upload this to your GitHub project repository.

Data Dictionary

A data dictionary is a documented collection of data about the database and each table and field within, including names, definitions, default values, attributes, acceptable values. This is important so that you document each table so you understand the various fields, data types, sizes, etc.

Example data dictionary for a table called "employee":

Table: Employee						
Field	**Type**	**Format**	**Size**	**Description**	**Required?**	**Key?**
empID	INT	999999	6+	Auto-generated unique identifier for each customer. First value starts at 100001	Y	PK
firstName	varchar(32)	Text	50	Customer First Name	Y	
lastName	varchar(32)	Text	50	Customer Last Name	Y	
startDate	date	YYYY-MM-DD		Date customer account was created. Default: Today	Y	
street	varchar(32)		50		N	
city	varchar(32)		50		N	
province	varchar(32)		2	Link to province table. Default: NB	Y	FK
postcode	varchar(32)	L9L9L9	6	Only store alphanumeric values. Display as L9L 9L9	N	
phone	varchar(32)	5065551212	9	Only store numeric values. Display as (506) 555-1212	N	
email	varchar(32)	x@x.ca	50		N	
…						

Security

All employees using the system must have an active account and password (see employee table)

Password must be hidden on login form(s) and encrypted in DB (i.e. not stored in plain text) with AES 256 bit encryption

When viewed in the db, the password will be encrypted and NOT in plain text

Audit Activity

Every action in the system creates a record of that activity in the Audit table, so you need to develop this aspect of the system now so you don't have to backtrack later.

Info to be tracked in each audit record includes:

- txnAuditID – unique, autonumbered ID
- txnID – id of the transaction being audited (if applicable)
- txnType – type of transaction (if applicable)
- siteID – site where activity originated
- employeeID – employee who initiated the activity
- status – status of the transaction (if applicable)
- date/time – current datetime when this record is created
- description of the activity – a clear text description of what is happening
    - Examples:
        - Add new employee record jsmith
        - Edit employee record jsmith

A trigger or an audit object that takes in the parameters above and writes them to the txnAudit table in the DB. However you do it, remember: ***it needs to be available for any transaction or activity that occurs in the system***. Think of this as the logs, breadcrumbs required to trace activity, and a way to any potential issues or improper activity.

User Permissions

Admin requires the ability to set and remove user permissions in the system.

Methods:

- Position-based assignment
  - Each new user created gets assigned the appropriate permissions required for the position they are assigned (see ***Add User***, ***Edit User***)
- Individual permissions
  - Each user can have additional permissions (beyond those assigned to their role)

Actor(s): ADMIN

Permission(s): SETPERMISSION

Sample: None

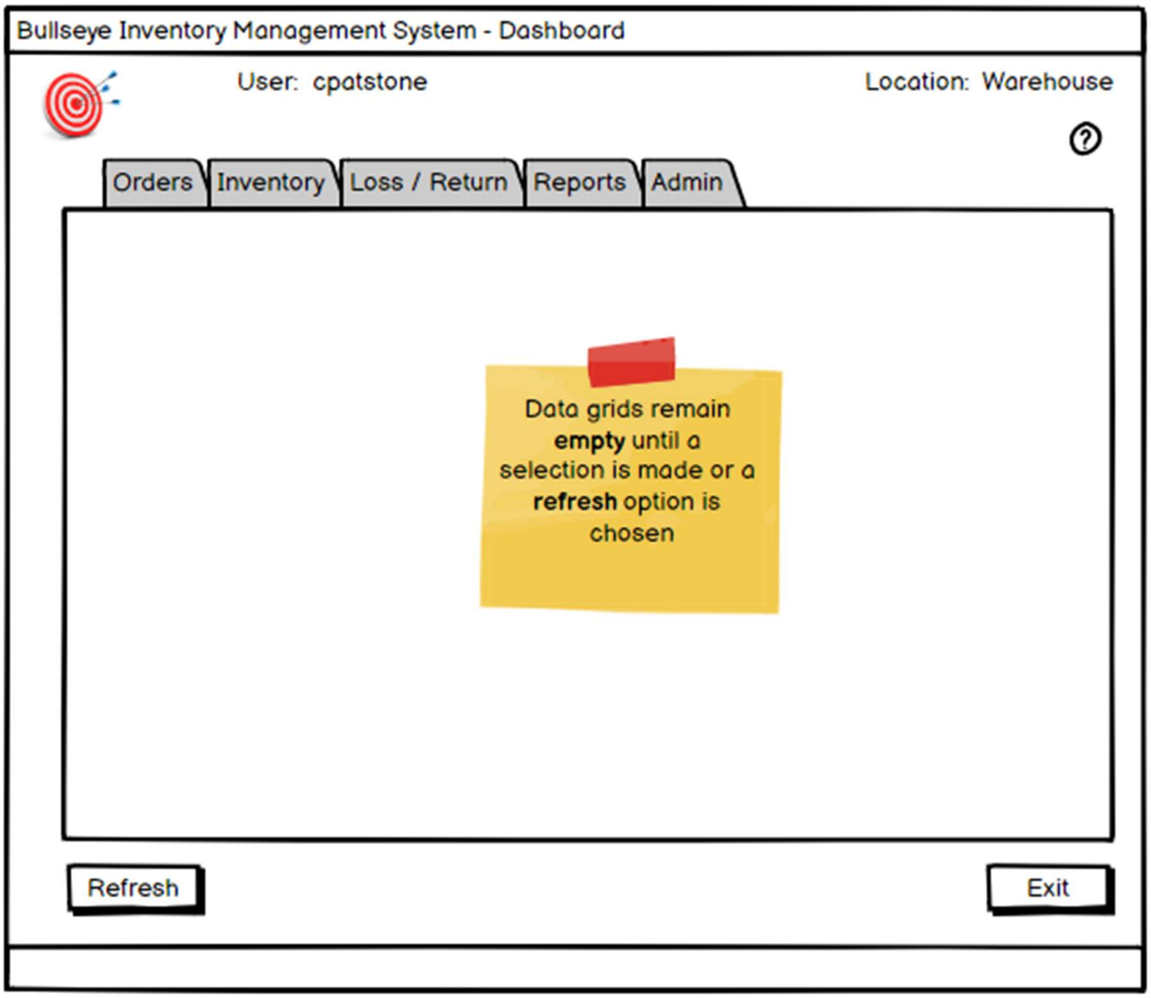
Dashboard

A dashboard that provides access to all aspects of the system, based on who is logged in.

Actor(s): ALL

Permission(s): ALL* (access to various areas/functionality is determined by permissions)

Sample:



Login

Every Bullseye user must have:

- A valid Username and password (first initial, last name - i.e. jperez, as described in the DB)
- Ability to recover username and/or password via email (we will mimic this with a form like the one below)
- permissions to perform specific actions
- Password encrypted in DB (i.e. not in plain text) – AES 256 bit
- Password rules: minimum 8 characters, start with a letter, contains at least 1 capital letter and 1 special character

A user who logs in with the correct username and password will be presented with the dashboard and have access to the appropriate functionality.

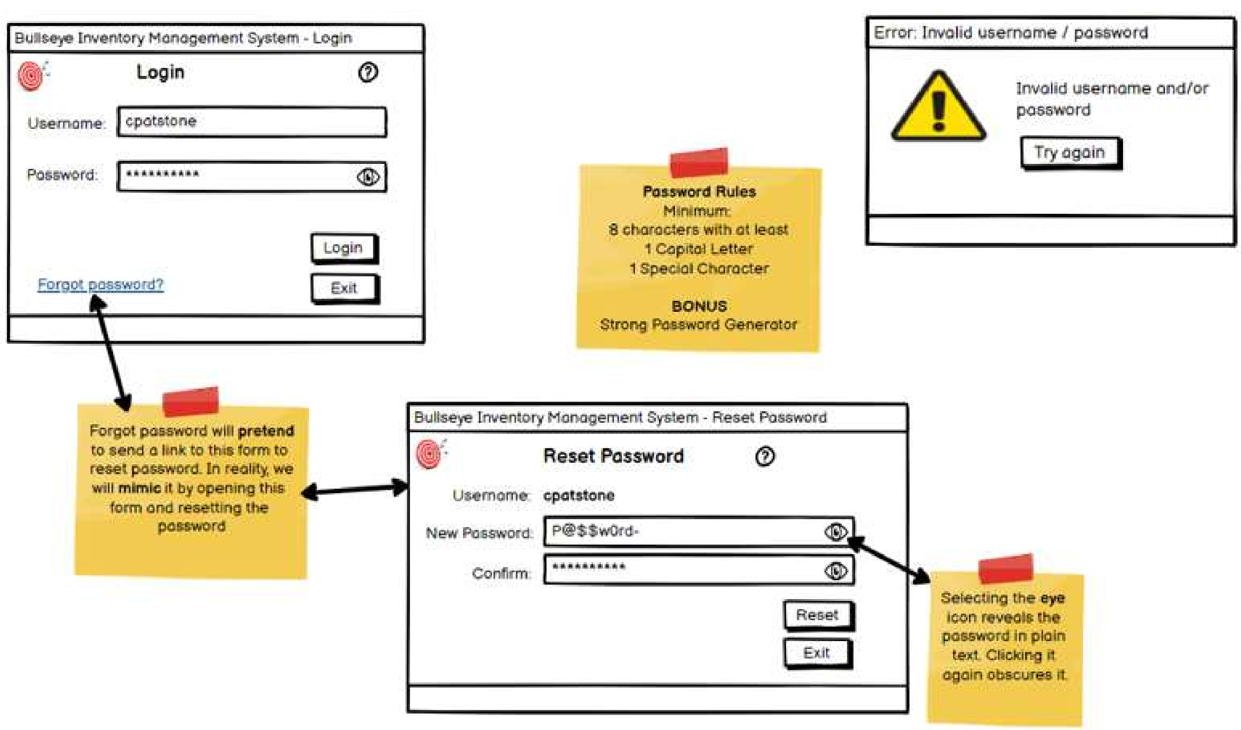
Any attempt to log in with an account that is inactive should fail (with appropriate warning)

Any attempt to log in with a username/password combination that is not accurate should fail (with appropriate warning)

Actor(s): ALL

Permission(s): None * (all active users accounts can log in)

Sample:



Logout

Every active Bullseye user can logout if they are currently logged in.

Logout methods:

1. Manual (select 'Exit' button from dashboard)
2. Automatic (after a period of inactivity, a time value which can be modified, default is 15 minutes). You can use either a config file or a database table to set this value.

Actor(s): ALL

Permission(s): None * (all active users accounts can log out)

Sample: None

Add User

Admin requires the ability to add a user to the system.

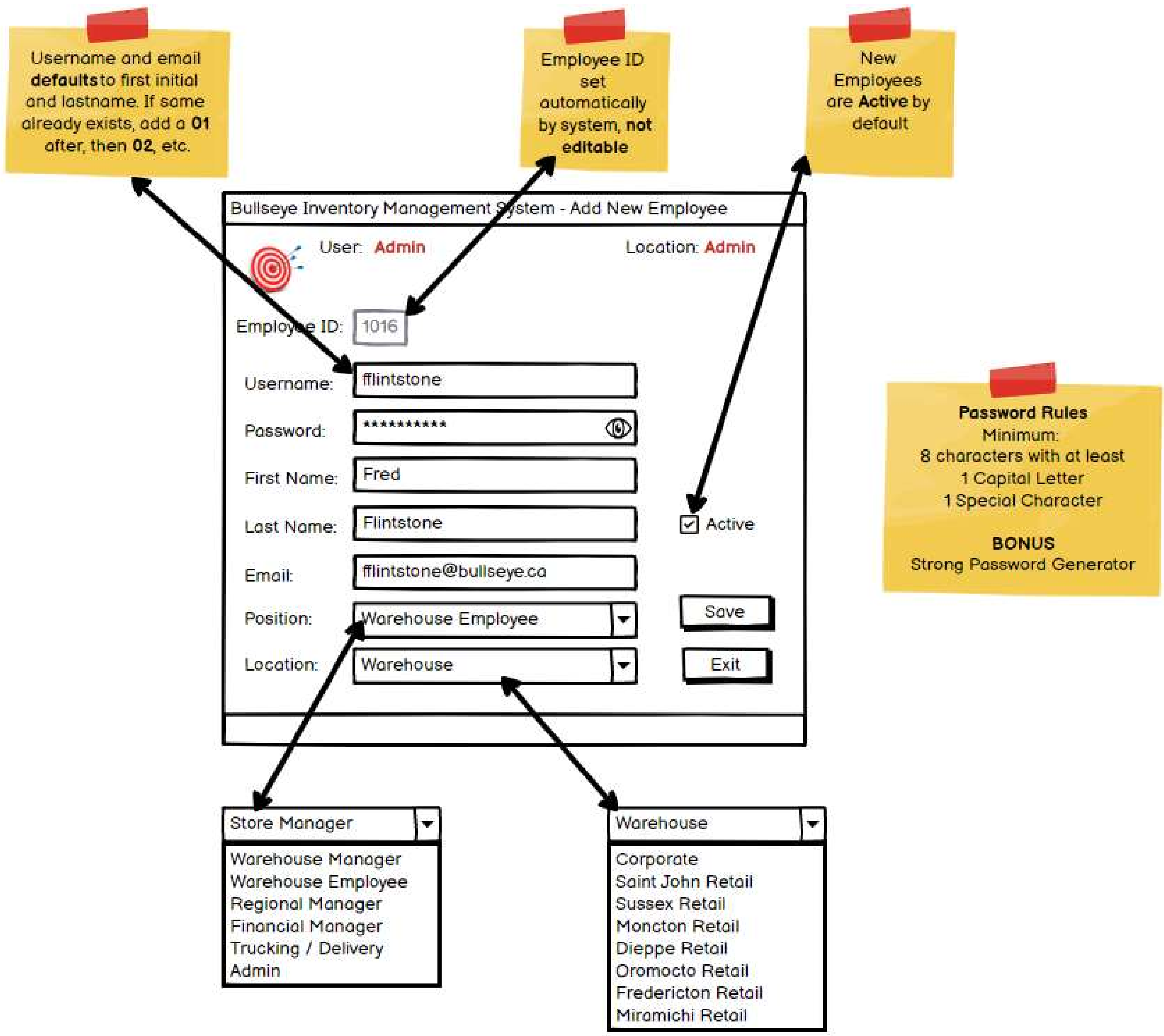
This will normally be triggered by a new user joining the company.

New user accounts are set "active" by default (the system sets their "active" status to TRUE or 1 in employee table).

Actor(s): ADMIN

Permission(s): ADDUSER

Sample:



Edit User

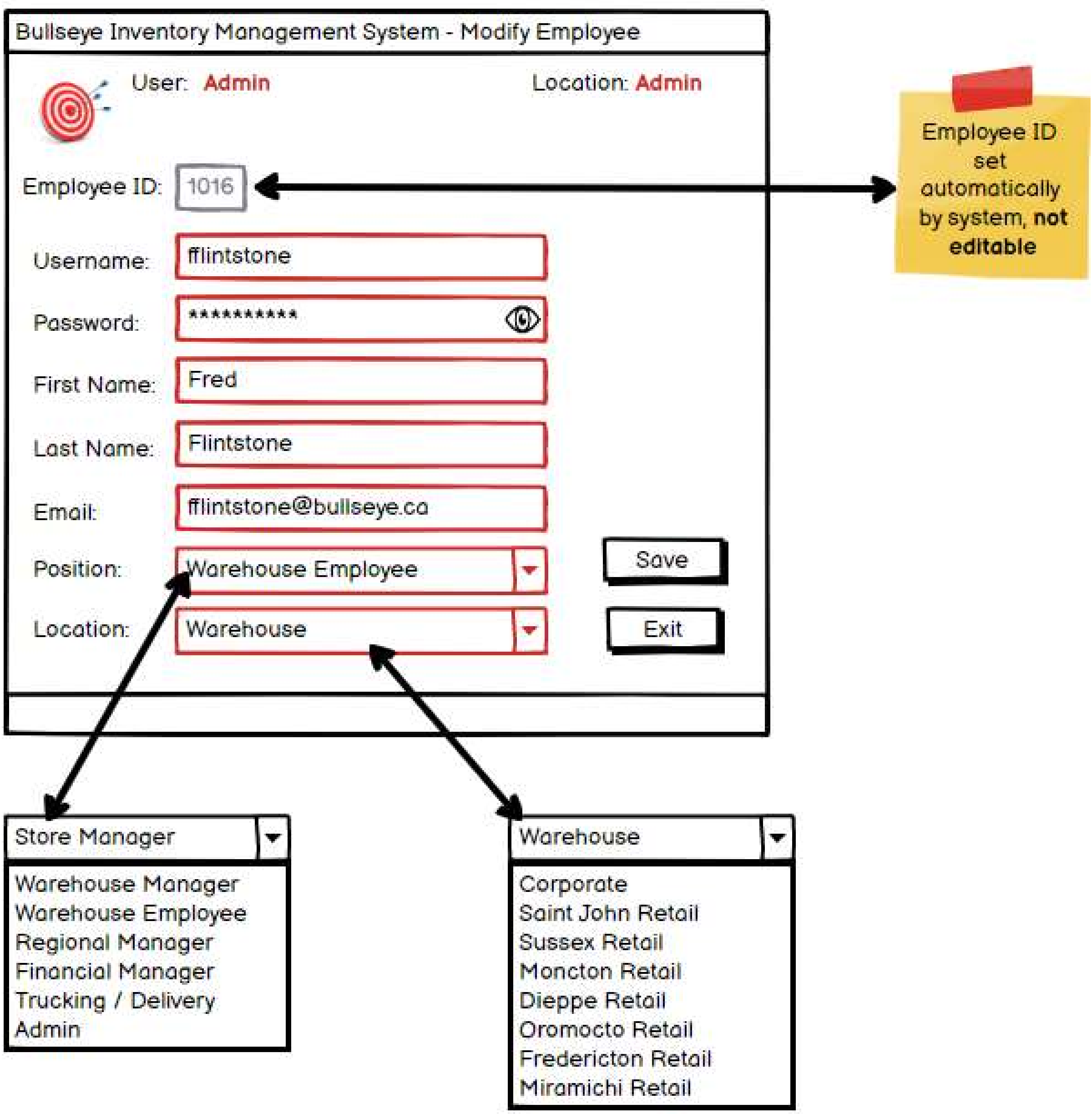
Admin requires the ability to edit a user's data in the system.

This will normally be triggered by a user changing jobs, getting married, etc.

Actor(s): ADMIN

Permission(s): EDITUSER

Sample:



Delete User

Admin requires the ability to remove a user from the system.

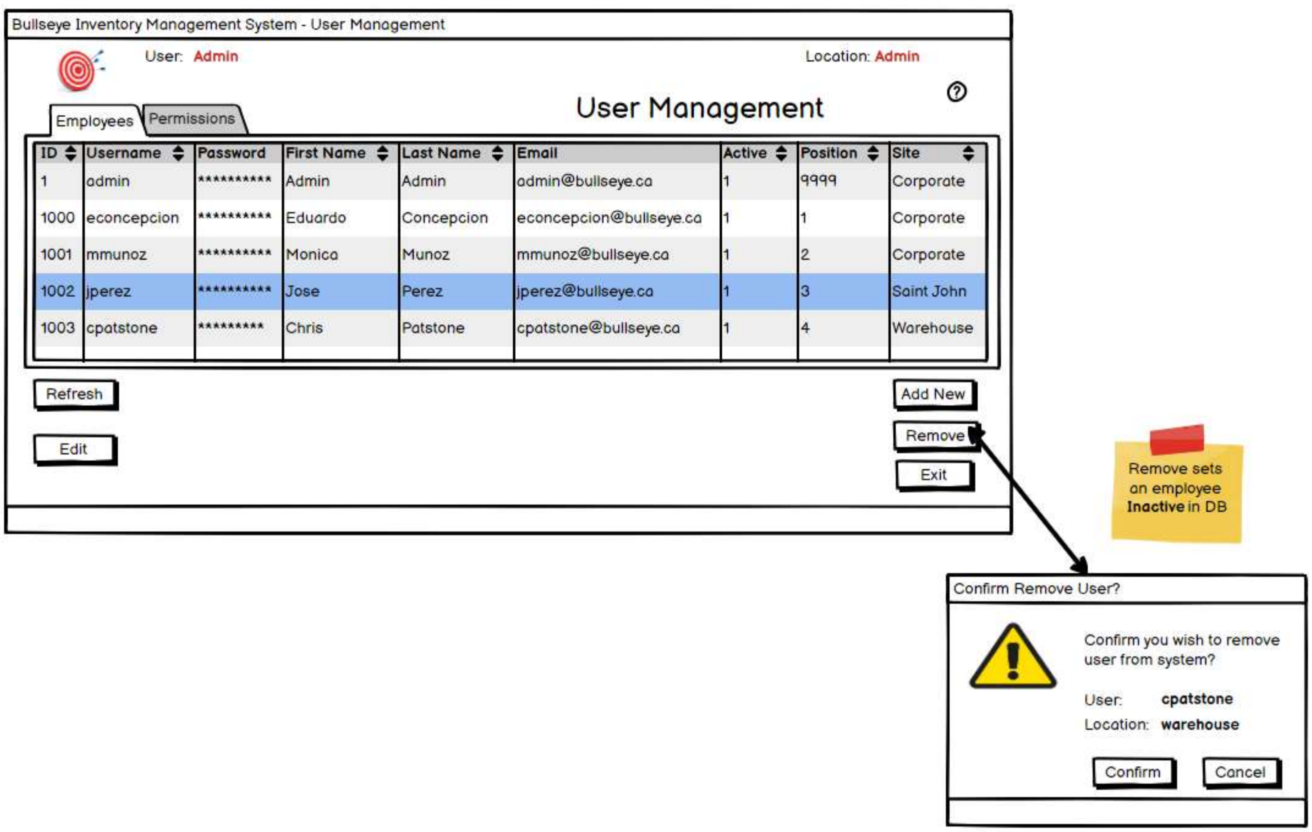
This will normally be triggered by a user leaving the company.

Any user “removed” is NOT deleted, the system simply sets their “active” status to FALSE (0 in employee table).

Actor(s): ADMIN

Permission(s): DELETEUSER

Sample:



Sprint 2

Create Store Order

Create a new weekly order for a store (site).

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

Features/Notes:

- 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

NEW Store Manager has created a new store order

SUBMITTED Store Manager has submitted the store order to the warehouse

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 over the reorderThreshold
- Store Manager has option to edit quantity of any item(s) being ordered before submitting, add any other items, remove any items, etc.(by case size)
- Can only add items which are ACTIVE = 1

siteIDTo (where the order will be delivered, i.e. the store)

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: ORDER

txnStatus: NEW or SUBMITTED

Permission(s): CREATESTOREORDER

Create Emergency Order

Create a new emergency order for a store (site).

*** Can (should) likely be couple with **Create Store Order** form

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

Features/Notes:

- A new emergency order can be created ONLY IF no other emergency order is currently open and active for this location/site
- Only 1 NEW order of type EMERGENCY can be active for each site (unlimited number in other statuses)
- Can only add items which are ACTIVE = 1

NEW Store Manager has created a new store order

SUBMITTED Store Manager has submitted the emergency order to the warehouse

Process:

- Store Manager can add up to a maximum of 5 separate line items

To (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)

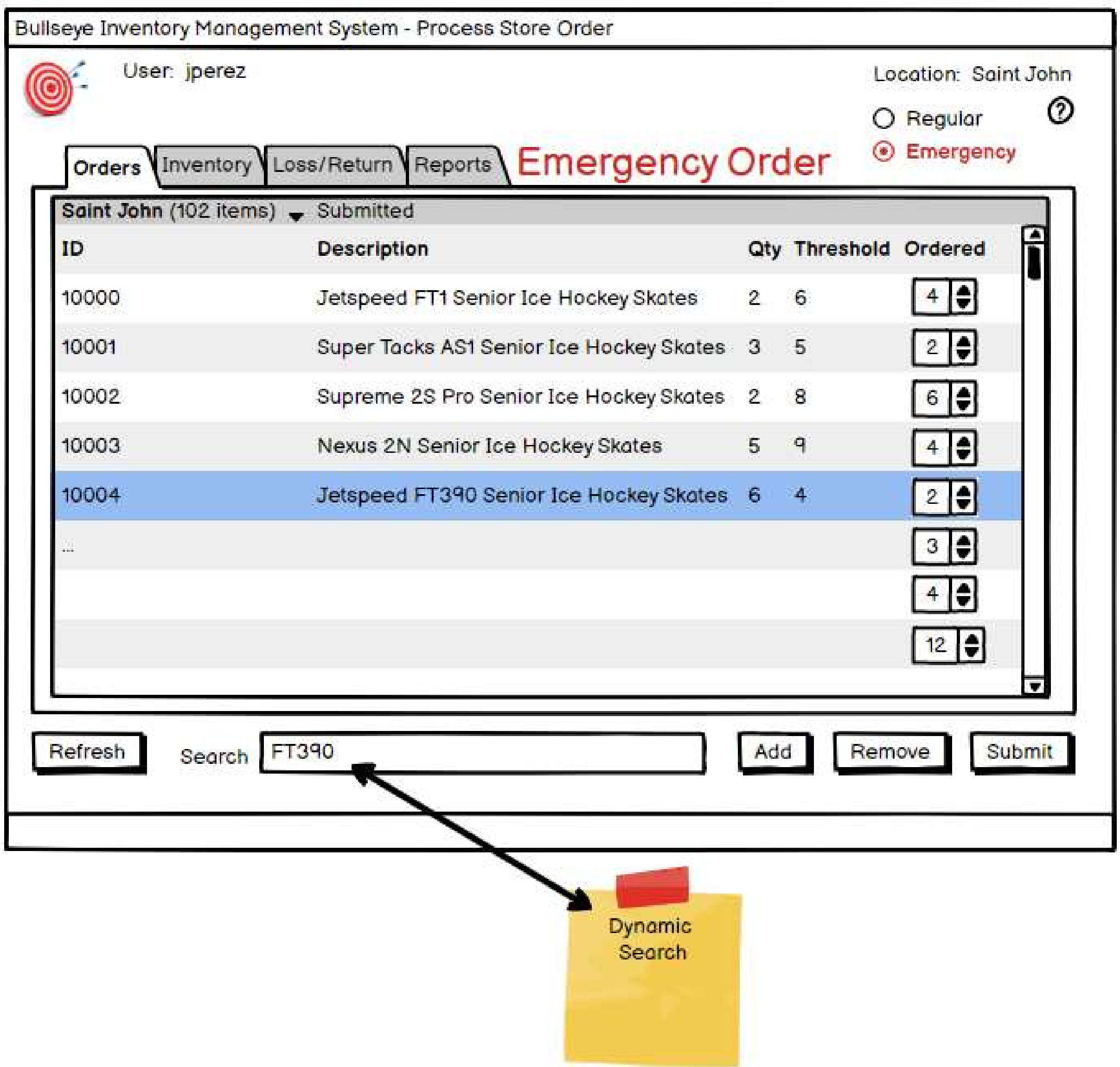
shipDate (should default to be the next regular delivery date for the siteIDTo location, then may be updated to be a different shipDate)

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

txnType: EMERGENCY

txnStatus: NEW or SUBMITTED

Permission(s): CREATESTOREORDER



Receive Store Order

Actor(s): Warehouse Manager

Description: Warehouse Manager receives an order from store site

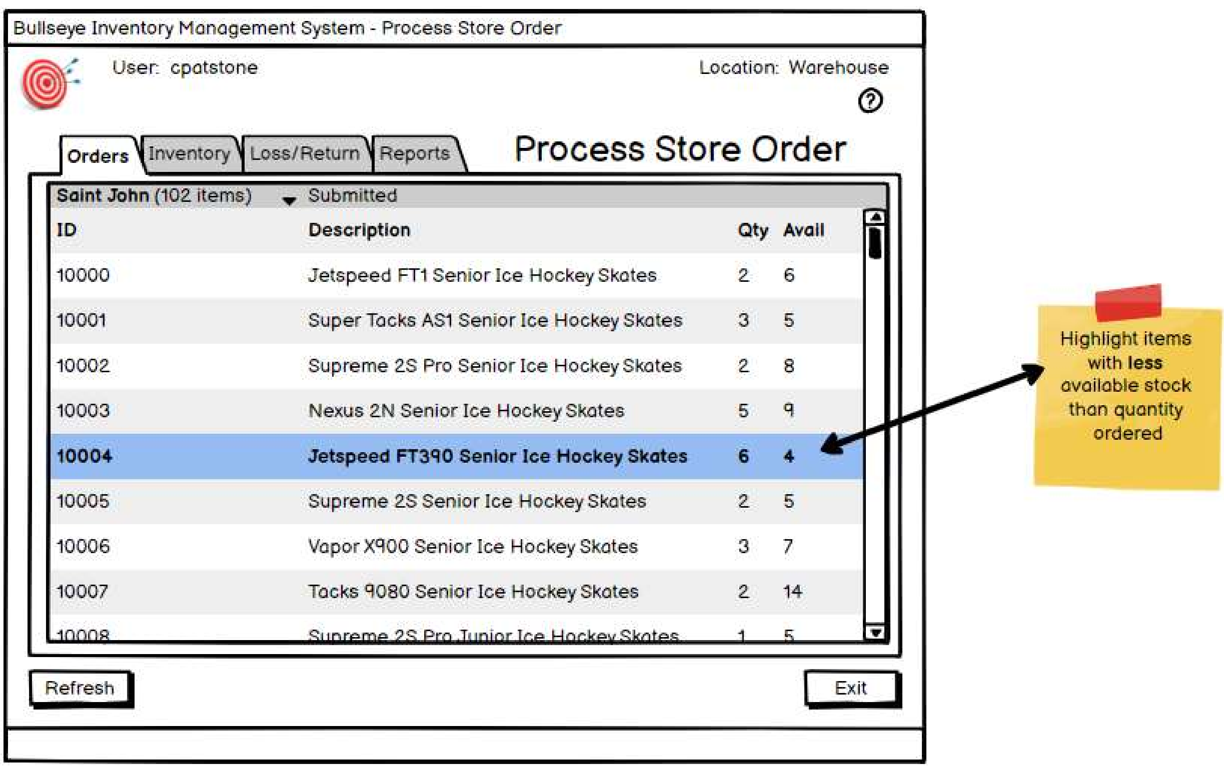
Features/Notes:

txnType: ORDER, EMERGENCY

txnStatus: SUBMITTED -> PROCESSING

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

Permission(s): RECEIVESTOREORDER



Fulfil Store Order

Actor(s): Warehouse Manager, Warehouse Workers

Description: Complete order for a store (site).

Features/Notes:

This is the process of warehouse workers manually checking off items

Orders should be fulfilled by making the warehouse employee check off each item as it is gathered for the order until it is complete, at which time the txnStatus = READY

Items added to the order should be removed from the warehouse inventory and added to the warehouse bay inventory (siteID=<ordering_siteId>, itemLocation="2" (Warehouse Bay)

txntype: ORDER, EMERGENCY

txnStatus:

- PROCESSING while order is being prepared by warehouse workers
- READY once order is assembled at warehouse and ready for pickup

Permission(s): FULFILLSTOREORDER

Add Item to Backorder

Actor(s): Warehouse Manager, System

Description: Add an ordered item to a backorder

Features/Notes:

- 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

Permission(s): CREATEBACKORDER, VIEWORDERS

View Store Order

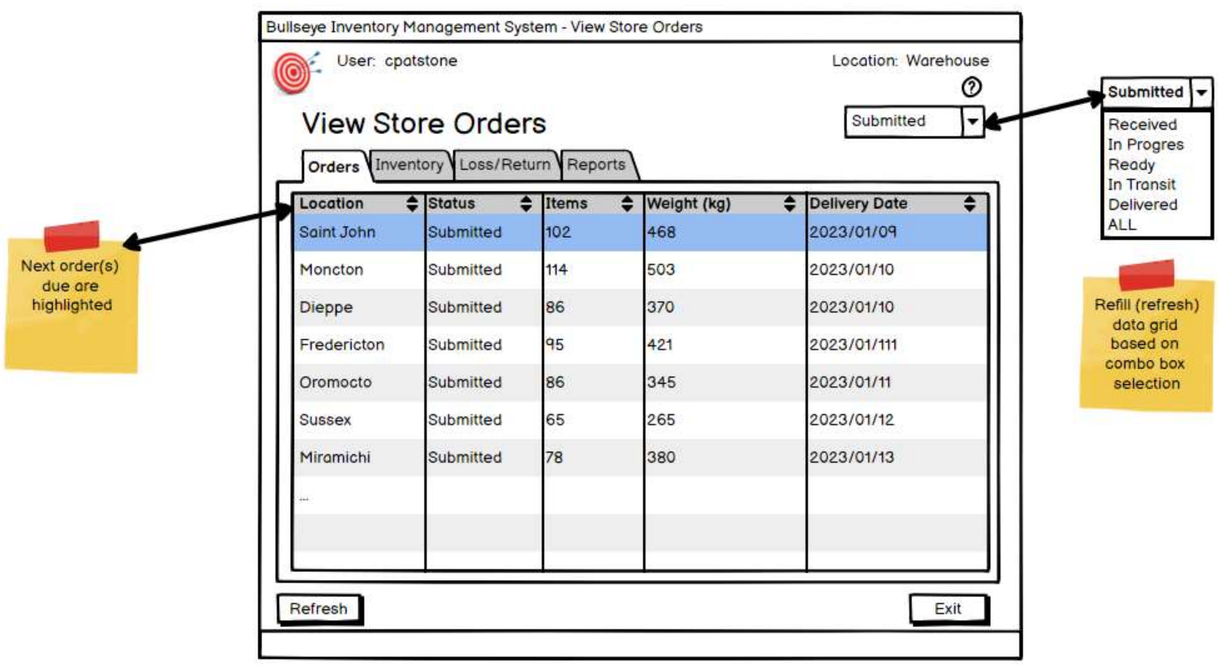
Actor(s): Warehouse Manager, ANY Store Manager, ANY Asst Store Manager

Description: View all outstanding store orders

Features/Notes:

- View all outstanding store orders (those with a txnStatus NOT "CLOSED")
- Default to show orders only from user's site
- Each Store should be able to see ALL their own orders,
- Warehouse Manager should be able to see ALL orders
- 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

Permission(s): VIEWORDERS



View Location

Actor(s): ALL

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

Features/Notes: All Users (including any customers online): Read only

Permission(s): None required

Add Location

Actor(s): Admin, Warehouse Manager

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

Features/Notes: Admin can add new sites as the company grows

Permission(s): ADDSITE

Edit Location

Actor(s): Admin

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

Features/Notes:

Permission(s): EDITSITE

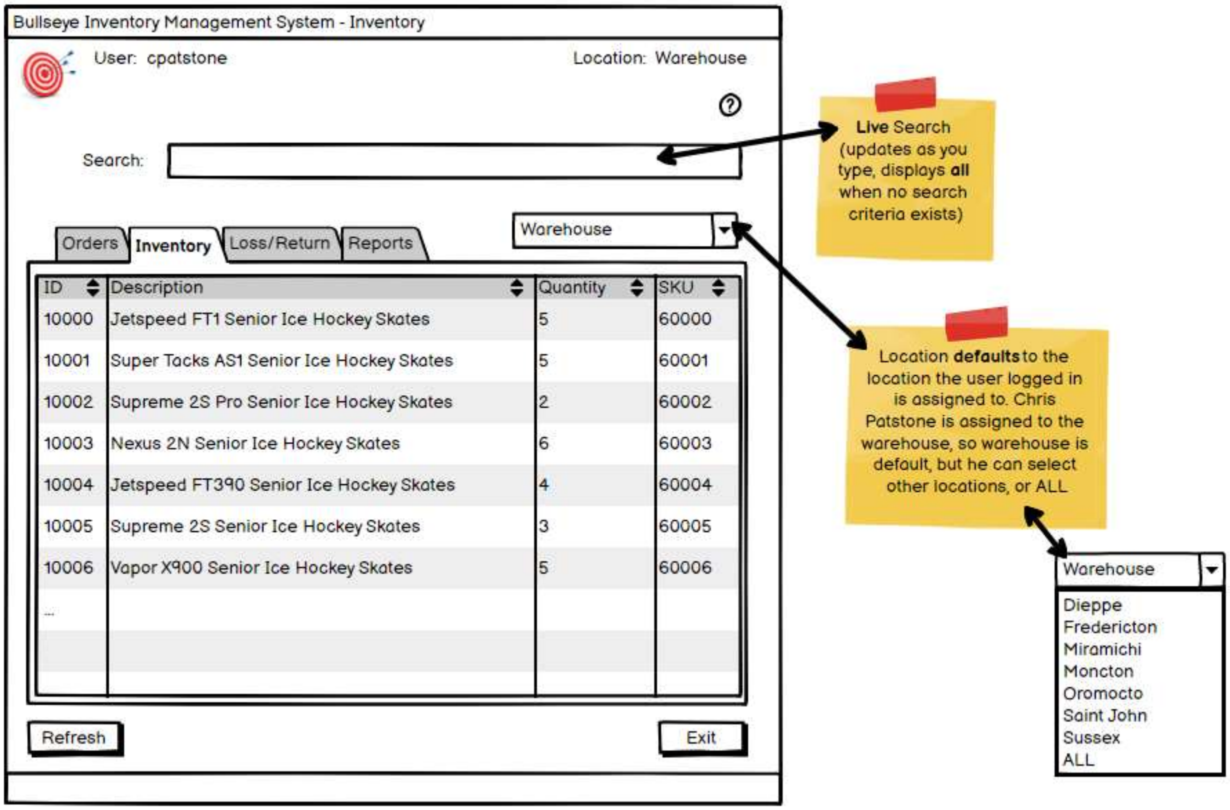
View Inventory

Actor(s): ALL

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

Features/Notes: All Users (including any customers online): Read only

Permission(s): None required



Edit Inventory

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

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

Features/Notes: Store Managers or Warehouse Manager can only change the **reorderThreshold** for their own location - i.e. Store Manager of Moncton retail store can modify the **reorderThreshold** for any inventory item in the Moncton store only

Warehouse Manager can set the item ACTIVE status

Permission(s): EDITINVENTORY

Move Inventory

Actor(s): All

Description: Used whenever inventory changes "place"

Features/Notes:

Change location of inventory (i.e. warehouse-to-order, storage to shelf, shelf-to-shelf, etc.)

Examples:

- Inventory is placed into an order (removed from the Warehouse inventory and associated with an order, siteID=<warehouse_bay>, itemLocation = <orderid>)
- Inventory is loaded on a truck for delivery (removed from warehouse_bay 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

Permission(s): MOVEINVENTORY

Sprint 3

Pickup Store Order

Actor(s): Driver, Warehouse Workers

Description: Driver Picks up order for a store location.
Items transferred to location=TRUCK (9999)
Order Types: ORDER, EMERGENCY
Status: READY -> IN TRANSIT

Features/Notes:
NEED to account for weight to identify TRUCK size for ALL orders going out on a particular day
NEED Web page for Acadia to open and view all upcoming deliveries so they can prepare proper truck size ahead (HINT: show required truck size on this! 😉)

Permission(s): DELIVERY, MOVEINVENTORY

Transport Store Order

Actor(s): Driver

Description: Transport order to a store location

Features/Notes:
Order Types: ORDER, EMERGENCY
Status: IN TRANSIT
Item location=TRUCK (9999)

Permission(s): None required

Deliver Store Order

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

Description: Deliver order to a store location

Features/Notes:
Order Types: ORDER, EMERGENCY
Status: DELIVERED
Item location=TRUCK (9999)

Permission(s): DELIVERY, MOVEINVENTORY, ACCEPTSTOREORDER

Accept Store Order

Actor(s): Store Worker, Store Manager

Description: Receive an order from delivery driver or courier
Features/Notes: Order Types: ORDER, EMERGENCY
Status: DELIVERED -> CLOSED
Item location=siteID
Update quantity in this site to reflect new items received
Some kind of checklist (mimicking a scanning process) to accept items on an order into the site

Permission(s): ACCEPTSTOREORDER

Place Online Order

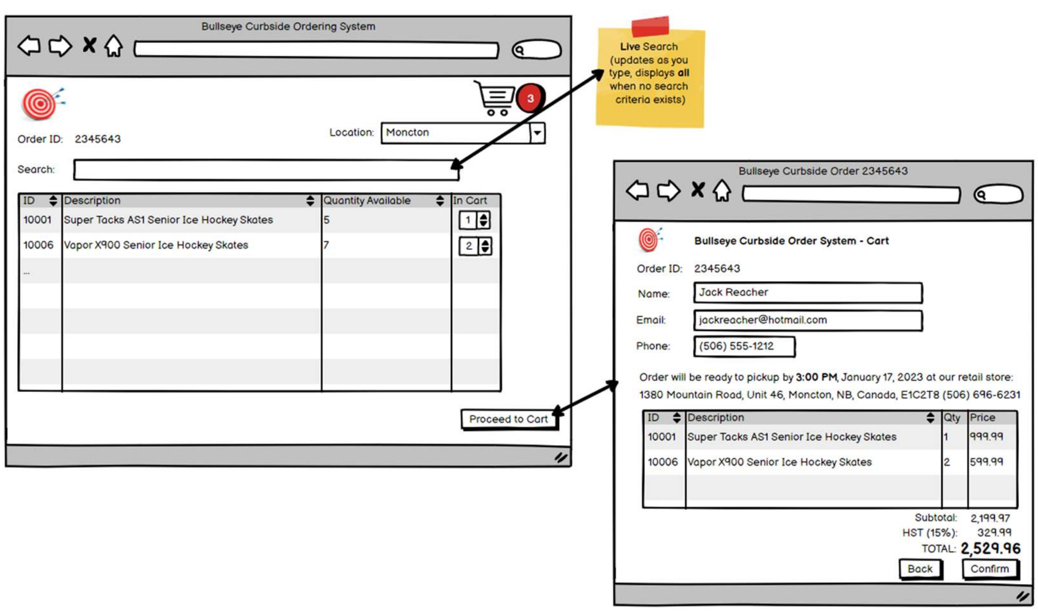
Actor(s): Customer (no account required)

Description: Select a store - ORDER from specific store, including displaying inventory amounts

Features/Notes:
Transaction type = ONLINEORDER
txtStatus = PROCESSING
txnType = Curbside

Customer can only order items that are in stock at the selected store location

Permission(s): None required, customers submit name, email address, and phone number. Just add a couple of text boxes to the web page that capture name, phone, and email, then add that info to the Notes field in the transaction. Customer should be able to come back to the website and search for order status by txnID, email.



Prepare Online Order

Actor(s): Store Manager, Customer

Description: Store receives an online order
Store Manager (for the associated store) sees new online orders for their store
Store Manager (for the associated store) fulfils online orders for their store

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
Item inventory quantity needs to be deducted

Features/Notes:
Use same statuses as for store orders, except no requirement for 'IN TRANSIT' status
Transaction type = ONLINEORDER
Item Location = Curbside (11)
txtStatus = PROCESSING -> READY
txnType = Curbside
Permission(s): None required

Receive Online Order

Actor(s): Store Manager, Customer

Description: Curbside pickup at appropriate store when status = READY
Customer can enter their order id (txnID) OR email address online and see the current status of the order.

Features/Notes:
STATUS: READY -> CLOSED once delivered curbside
Checklist of some sort to mimic customer signing

Permission(s): None required

Modify Record*

Actor(s): ADMIN
Description: Modify data in ONLY the following tables (i.e. CRUD items in tables for ADMIN only - NOT DELETE records)

- Transactions
- Orders
- Backorders

ONLY FOR ***non-closed*** orders/transactions
Essentially to cancel an order
No need to transfer stock.

Basically anything in NEW, SUBMITTED, RECEIVED, PROCESSING status (anything before READY status)

Features/Notes: txnstatus = CANCELLED if you want to 'delete' a txn. Never actually delete
Permission(s): MODIFYRECORD

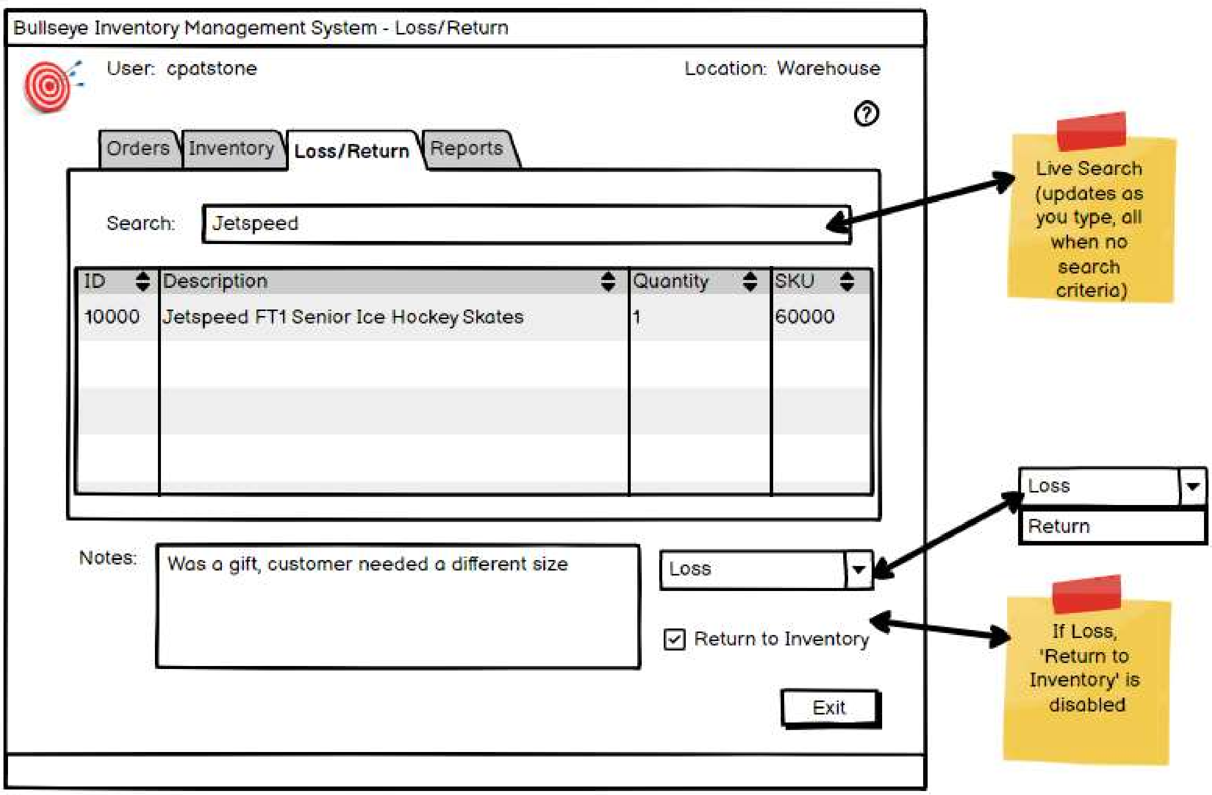
Sprint 4

Create Loss

Actor(s): Store Manager, Warehouse Manager

Description: Create a record to track lost inventory(or damaged inventory). Create an explanatory note in the txn record. The item is removed from the inventory for the location involved

Transaction Type: LOSS, DAMAGE
Features/Notes:
Permission(s): CREATELOSS



Process Return

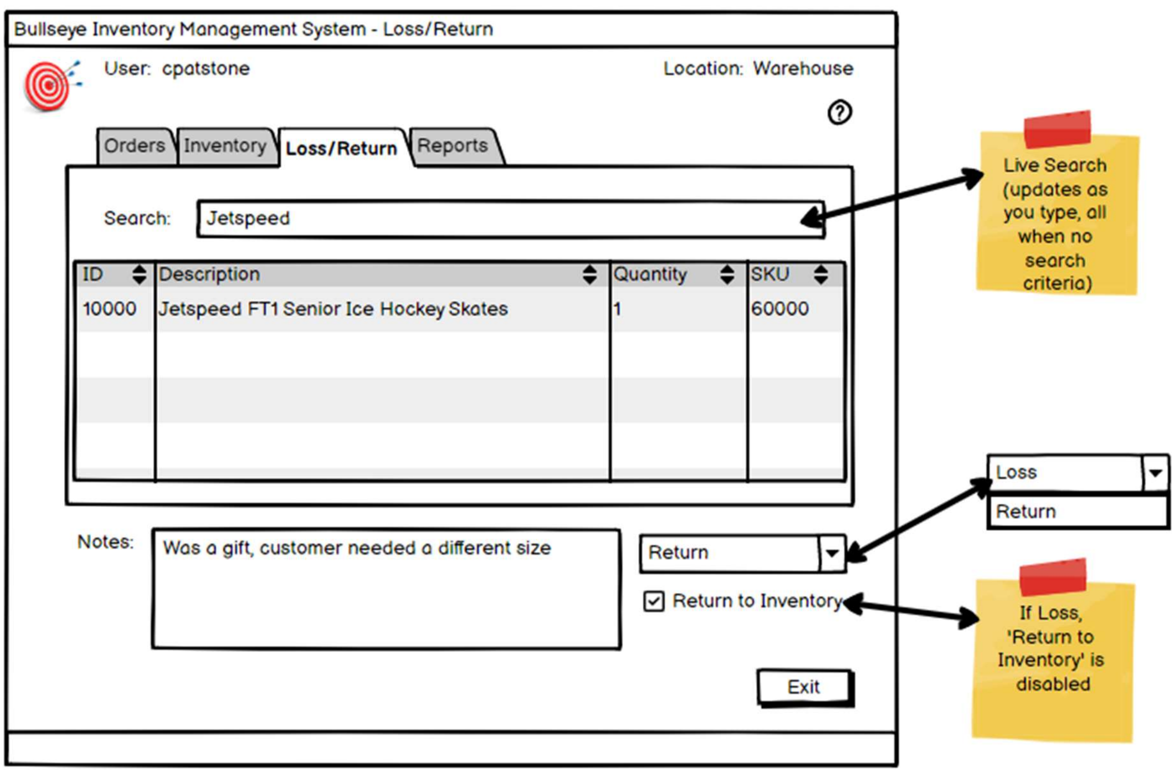
Actor(s): Store Manager, Warehouse Manager

Description: Create a record to track returned inventory. Create an explanatory note. The item is added to the inventory for the location involved

Transaction Type: RETURN

Features/Notes: Product is either added back to inventory if in good condition, or a LOSS is created if not in good condition

Permission(s): PROCESSRETURN



Add New Product

Actor(s): WAREHOUSE MANAGER
Description: Add new supplier product to **item** and inventory tables
Ensure when a new item is added, warehouse gets a quantity of **25** and a reorderThreshold of **25**, and each store has an inventory record created with a quantity of **0** and a reorder threshold of **5**
Features/Notes: Status: ACTIVE by default
Permission(s): ADDNEWPRODUCT

Edit Product

Actor(s): Warehouse Manager
Description: Edit inventory item information in item table (name and description, case size, weight, etc.)
Features/Notes: Delete/remove product = INACTIVE
Permission(s): EDITPRODUCT

Create Supplier Order

Actor(s): Warehouse Manager
Description: Inventory items automatically added when below minimum threshold (***similar to store orders***)

Features/Notes: Can also be done manually.
Can be sorted/filtered by supplier (each supplier's item separated, the warehouse would use this to create the accompanying report 'Supplier Order' under CREATE REPORTS use case.
Status types: NEW, SUBMITTED, DELIVERED, CLOSED
ONLY one order NOT in closed status can be open at any time
Permission(s): CREATESUPPLIERORDER

Create Reports

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

Description:

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 and mileage. DOES NOT NEED to have full order details of items. On a selected date

Store Order - list report for each individual store order, used as a record for store

Shipping Receipt - for each individual order, used as a record by driver, potentially a way to check items on and off, depending on how you built your system

Inventory - sort by ALL or by individual store, sorted by site. Have item names, not just IDs

~~**Sale** - sort by ALL or by individual store~~

Orders - sort by ALL or by individual store (Summary report, not each product detail) within a given date range

Emergency Orders - sort by ALL or by individual store, within a given date range

Users - list all, sort by role, location. Use role names, etc., instead of just position IDs

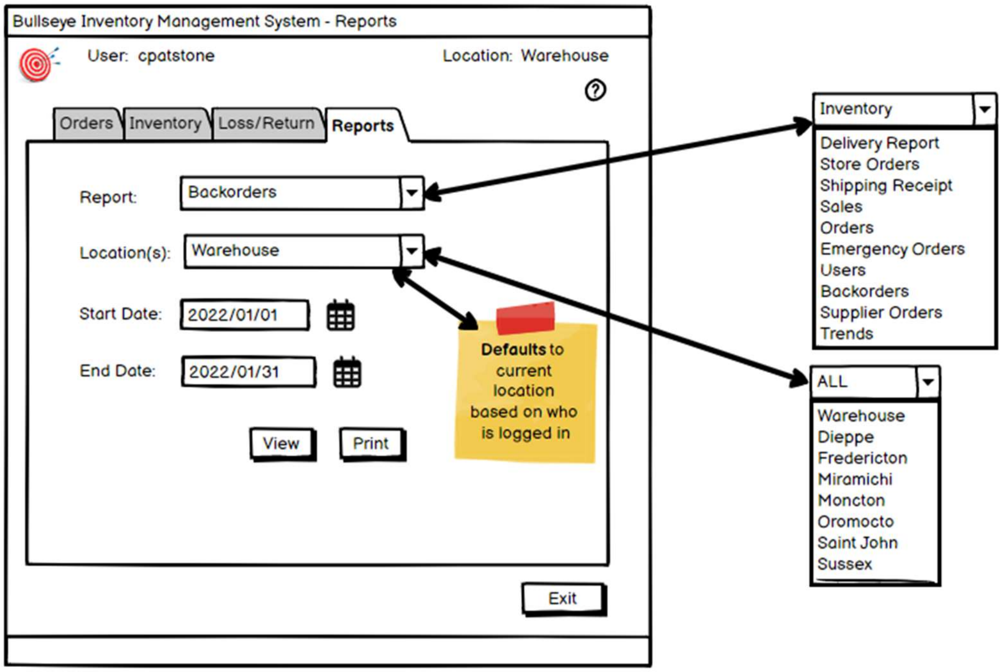
Backorders - sort by ALL or by individual store (include product detail), within a given date range

Supplier Order - (sort/filter by and/or page break between suppliers, used to send to the supplier to order more items), within a given date range. See above User Story

Loss/Damage/Return - sort by ALL or by individual store, sort/filter by and/or page break between Loss/Damage/Return, within a given date range

Features/Notes: ALL reports should allow for dynamic date ranges (i.e. datetimepickers for start and end dates for a time frame). Use appropriate reporting tool (Tableau, MicroStrategies, Data Analysis tool, HTML, Crystal Reports, etc.)

Permission(s): CREATEREPORTS



Audit

Actor(s): System

Description: Every action taken creates a record of the activity in the Audit table.

Features/Notes: Info includes: Unique ID, user, date/time, description of the transaction

Permission(s): None required, system manages this

Transaction Status State Chart