CPSC 471

Database Project Final Report

Group 35

Contents

[0](#_Toc469422074)

[2 Introduction 2](#_Toc469422075)

[3 System Requirements 2](#_Toc469422076)

[4 Entity-Relationship Design 3](#_Toc469422077)

[5 Hierarchical Structure 4](#_Toc469422078)

[6 Relational Model Design 6](#_Toc469422079)

[7 Appendix A – User Manual 7](#_Toc469422080)

[7.1 Login page 7](#_Toc469422081)

[7.2 Employee-Level Functions 7](#_Toc469422082)

[7.2.1 Search Apparel 8](#_Toc469422083)

[7.2.2 View Apparel 9](#_Toc469422084)

[7.2.3 Sale 10](#_Toc469422085)

[7.2.4 Return 10](#_Toc469422086)

[7.2.5 Logging Out 11](#_Toc469422087)

[7.3 Supervisor-Level Functions 12](#_Toc469422088)

[7.3.1 Add Employee 13](#_Toc469422089)

[7.3.2 Delete Employee 14](#_Toc469422090)

[7.3.3 Edit Employee 15](#_Toc469422091)

[7.3.4 Search Employee 17](#_Toc469422092)

[7.3.5 View Employees 18](#_Toc469422093)

[7.3.6 Add Apparel 18](#_Toc469422094)

[7.3.7 Delete Apparel 19](#_Toc469422095)

[7.3.8 Add Stock 21](#_Toc469422096)

[7.3.9 Search Apparel 21](#_Toc469422097)

[7.3.10 View Apparel 22](#_Toc469422098)

[7.3.11 Sale 23](#_Toc469422099)

[7.3.12 Stolen 23](#_Toc469422100)

[7.3.13 Write Off 24](#_Toc469422101)

[7.3.14 Logging Out 25](#_Toc469422102)

[8 Appendix B – Relationship Listing 27](#_Toc469422103)

# Introduction

In this project we took on the problem of creating an inventory database system for a small clothing retailer. Having relied on traditional methods to manage their stock in the past, our client has outgrown their pen-and-paper approach and are now in need of something more robust. Therefore, we’ve designed an electronic inventory system that allows them to track, manage, and record damages related to their stock, as well as manage employee access to inventory information. With these tools in hand, our client is much better equipped to handle their inventory challenges as they grow and evolve.

The database can be reached at the web address:

<http://10.146.164.120/login.php>

We’ve set aside some login credentials to help with testing:

User-level account – EID: 54321 Password: password

Supervisor-level account – EID: 11220 Password: password

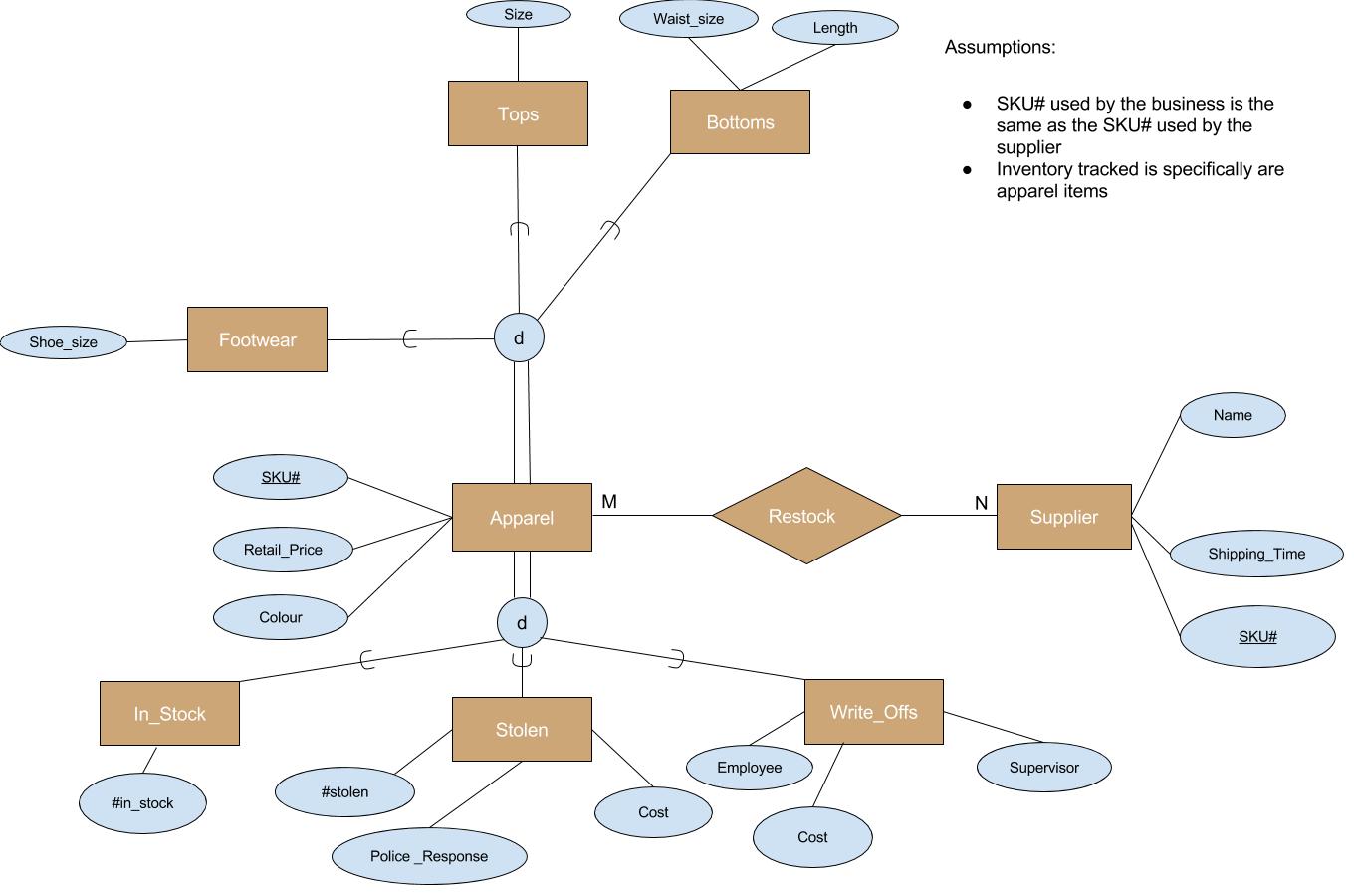
# System Requirements

The inventory system requires users to access a webpage in order to view any stock-related information. This requires the users to have access to a computer that has an internet connection and is able to run a modern web browser. The users must also have valid login credentials in order to gain access to the inventory system. These credentials will be created by a store supervisor and distributed to employees outside of the inventory system. Employees must also be able to operate basic computer peripherals such as a mouse and keyboard.

*Note: This webpage is designed to be accessed only from the University Of Calgary campus. Off-site access is only available over a VPN connection. Contact us for details.*

# Entity-Relationship Design

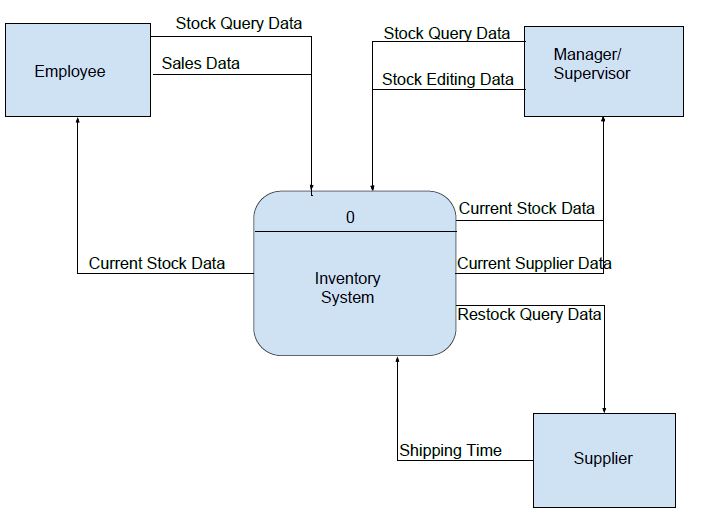
The conceptual design is largely based around the pieces of apparel themselves, with other extensions being built to give more functionality. The pieces of apparel are broken down into three sub-categories: footwear, tops, and bottoms. There are also sub-tables for recording stolen and written-off items, when needed. Additionally, supplier information is gathered and included when a user searches for an item that is out of stock, giving information on the supplier’s name and the estimated shipping time for replacement items.



**Figure 4.1:** Entity-Relationship Model

# Hierarchical Structure

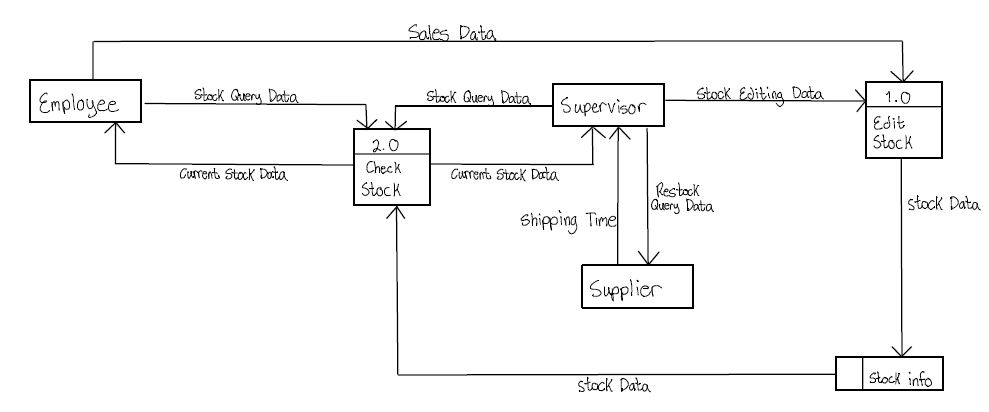
At the high level, the inventory system manages the interactions that employees, supervisors, and suppliers have with our client’s current inventory.



**Figure 5.1:** Context Diagram

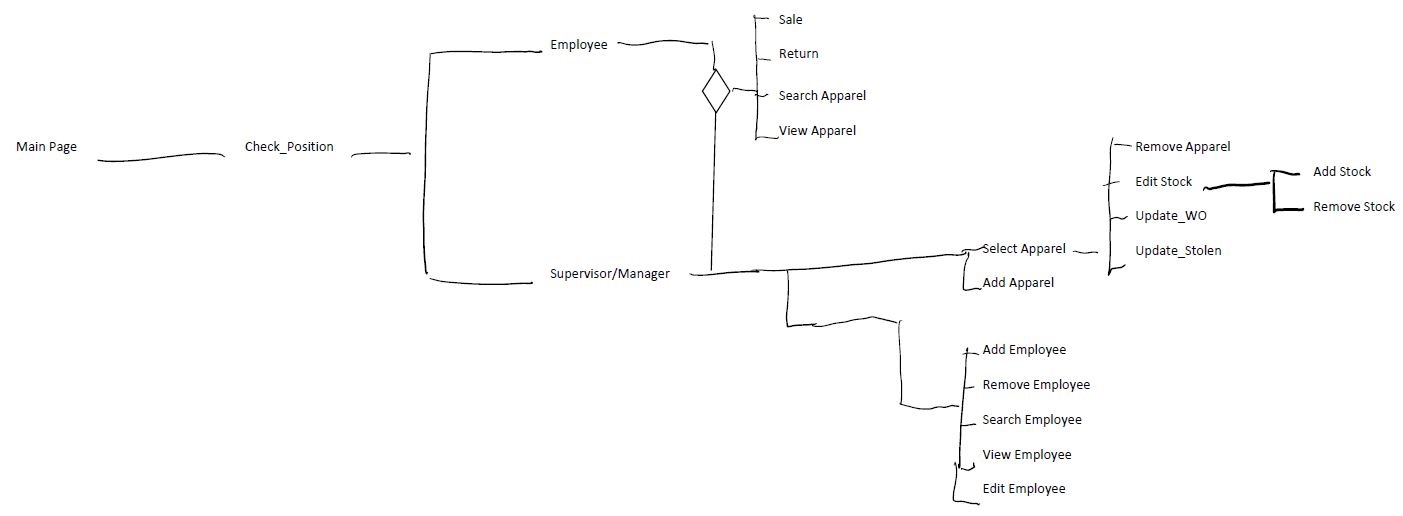
Employees are able to query for current stock data and view the results. Supervisors are able to query, create, edit, and view current stock data. Supervisors are also able to query, create, edit, and view current user account information. Suppliers are able to receive queries and return shipping information.

In the low-level design, the supervisor and employee entities are both able to access the Stock Check entity, while the supervisor alone has access to the Edit Stock entity. The supplier entity is prohibited from accessing either the Stock Check or Edit Stock entities directly.



**Figure 5.2:** Data-FlowDiagram Level-0

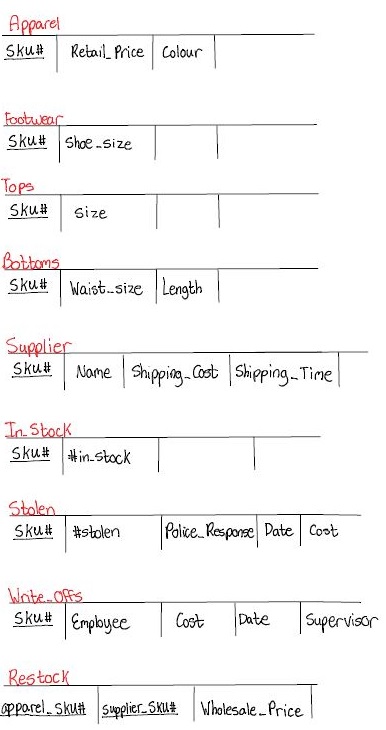
In terms of the user interface design, both the supervisor and the employee have access to the Sale, Return, Search Apparel, and View Apparel functions. However, only the supervisor has access menus for managing employee accounts and editing apparel information.



**Figure 5.3:** Hierarchy Input Process Output Diagram

# Relational Model Design

The relational model is designed to be a practical implementation of the Entity Relationship design. There are separate tables for the unique information regarding footwear, tops, and bottoms. There are also tables for stolen and written-off items. Supplier information is kept in a table, but it is not designed to be edited within the database; the information is populated from information given by the supplier.

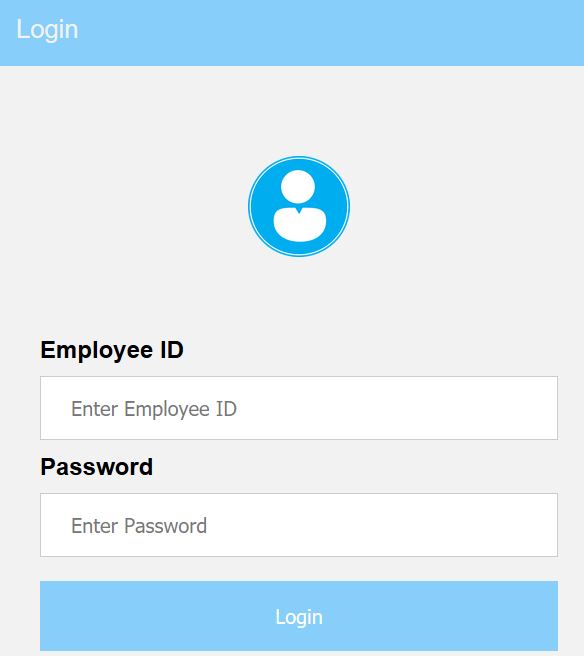


**Figure 6.1:** Relational Model

# Appendix A – User Manual

## Login page

When a user first visits the website they are greeted with a login page for them to enter their credentials.



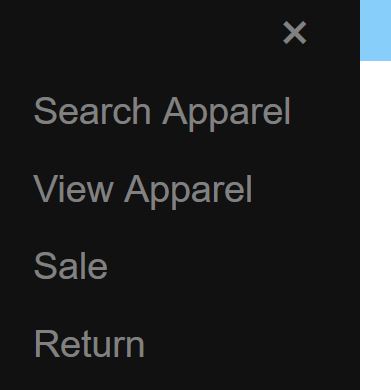
User credentials are created beforehand by a supervisor and distributed outside the inventory system. Employees do not have the ability to create their own user accounts.

## Employee-Level Functions

Once logged in with employee-level credentials, a user will be taken to the employee home page. Here they will be able to access all employee-level functions.

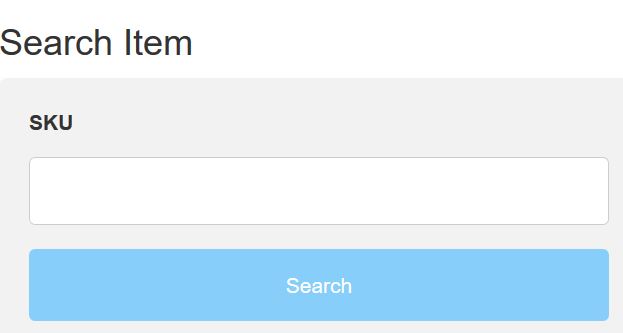


Clicking the ‘Menu’ button brings up a listing of employee-level functions.

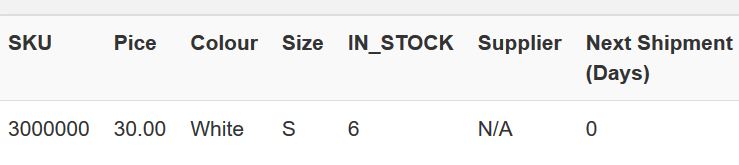


Selecting any of the available functions takes a user to that function’s page.

### Search Apparel



The Search Apparel function allows a user to enter a SKU# for an item of apparel and receive information about that item.



A user is able to receive information about price, colour, size, and stock level of the specified piece of apparel. If there are no items currently in stock for that piece of apparel supplier information is displayed of which supplier carries the item as well as the estimated shipping time of receiving fresh stock.



### View Apparel

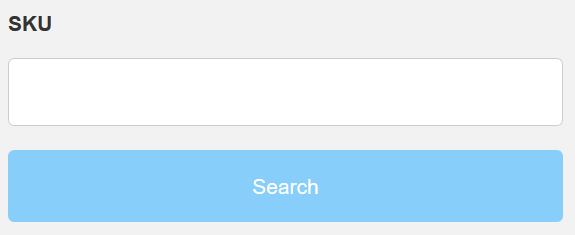
The View Apparel page allows a user to view all items of apparel currently in the database. A user will be able to view the price, colour, size, and number of items in stock.



All items are sorted by apparel type.

### Sale

The sale function allows a user to enter a SKU# for an item that is being purchased and have the inventory system adjust accordingly.

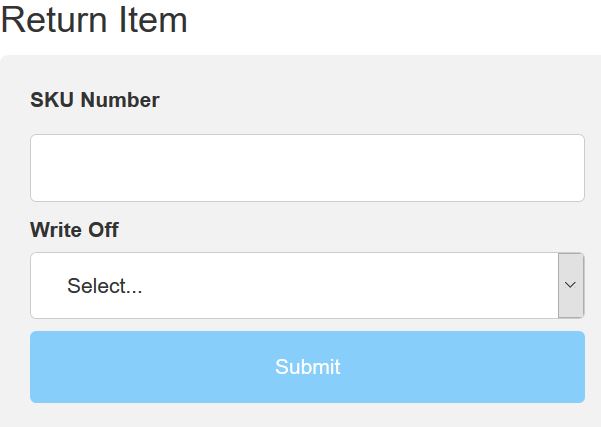


When the item has been purchased, the user is notified.



### Return

The Return menu allows a user to return an item of apparel. Entering the SKU# of the item allows the user to select the specific item to be returned. A user also has the option to choose to write-off the item in the return process. Choosing not to write-off the item adds the item back into the inventory database. Choosing to write-off the item adds the item to the write-off database and are not able to be sold later.



Once the item is successfully returned, the user is notified.



### Logging Out

When a user is ready to log out of the inventory system, they will first need to click the Options menu button in the top-right of the window.



This brings up the options menu, which includes a logout function.



Logging out takes the user back to the login screen.

## Supervisor-Level Functions

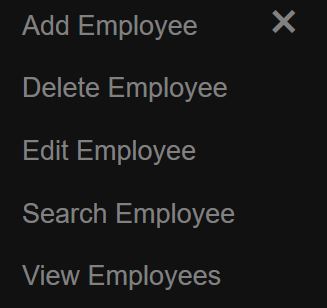
Once logged in with supervisor-level credentials, a user will be taken to the supervisor home page. Here they will be able to access all supervisor-level functions.



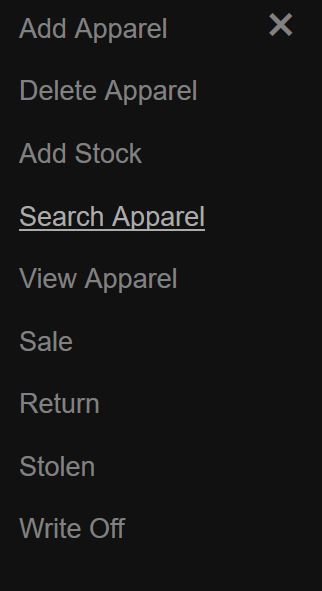
Clicking the ‘Menu’ button brings up a listing of supervisor-level functions.



Selecting the ‘Employee’ option brings up a menu of employee-management functions.



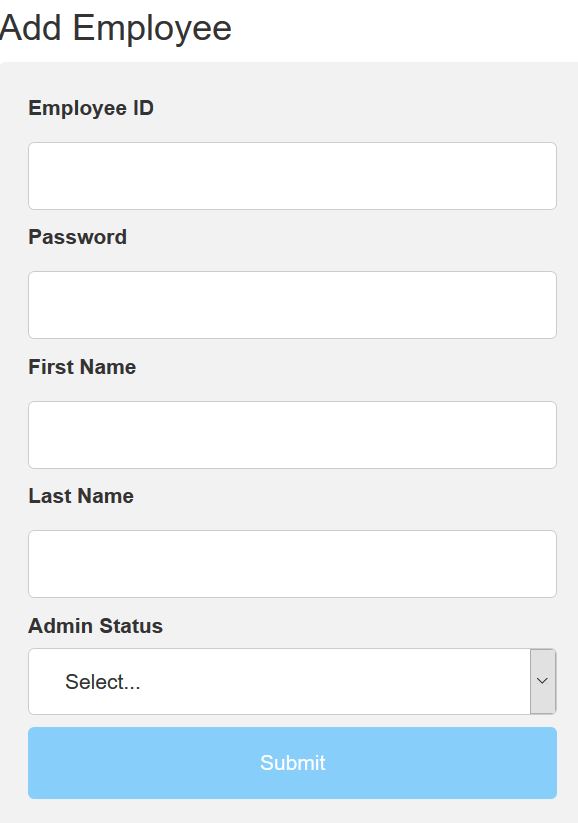
While selecting the ‘Apparel’ option brings up a menu of apparel-related functions.



Selecting any of the available functions takes a user to that function’s page.

### Add Employee

Selecting the ‘Add Employee’ function allows a supervisor to create new accounts for logging in to the database system.



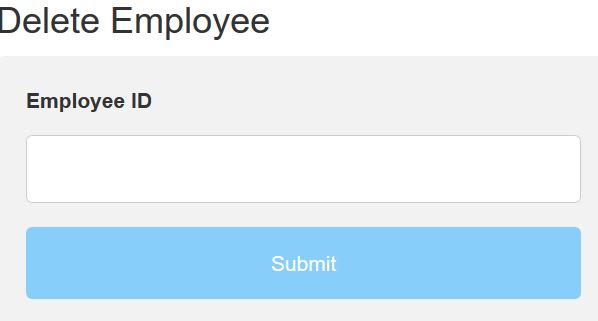
After inputting the new user’s employee ID number, password, first name, last name, and admin status, the supervisor clicks the ‘Submit’ button to create the account.



The supervisor is notified if the account has been successfully created.

### Delete Employee

Selecting the ‘Delete Employee’ function allows a supervisor to remove accounts for users logging in to the database system.



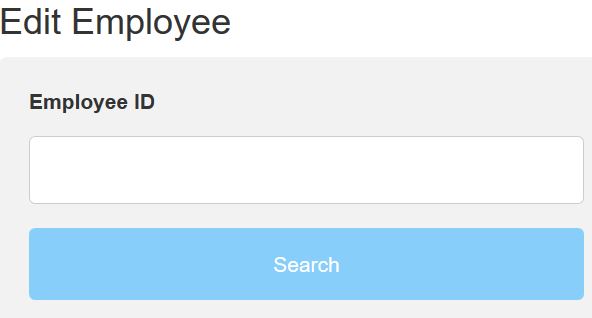
After inputting the desired user’s employee ID number the supervisor clicks the ‘Submit’ button to delete the account.



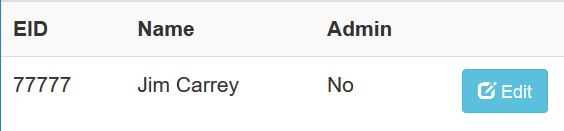
The supervisor is notified if the account has been successfully deleted.

### Edit Employee

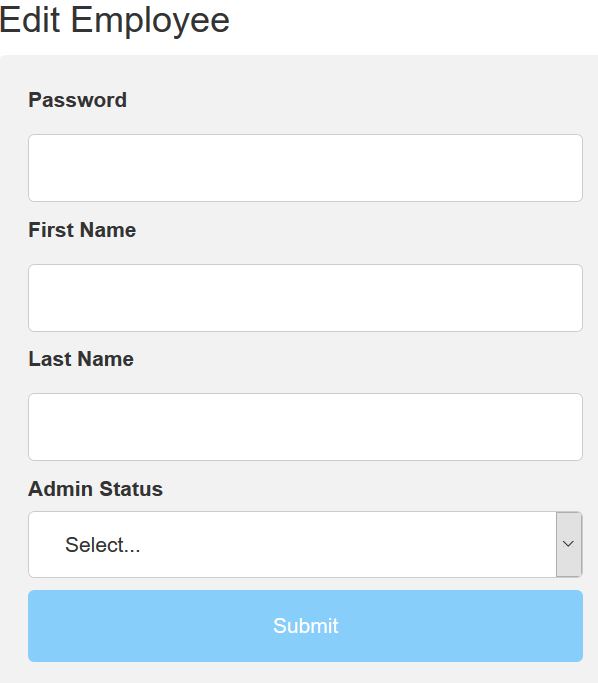
Selecting the ‘Edit Employee’ function allows a supervisor to modify the account information of users logging in to the database system.



After inputting the desired user’s employee ID number the supervisor clicks the ‘Submit’ button to begin view the current information for that employee.



If the supervisor wishes to change the current information, pressing the ‘Edit’ button takes them to a page that will allow them to change the information.



*Note: A supervisor cannot change a user’s Employee ID. The Employee ID is permanent once the user account has been created.*

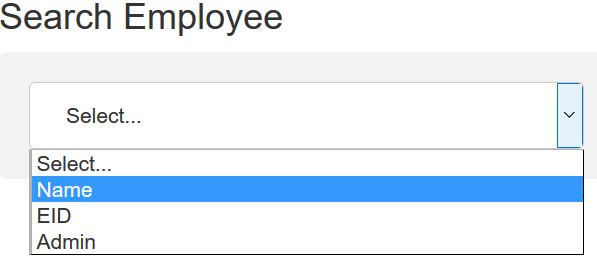
Here the supervisor can change the password, first name, last name, and admin status of the given user account.



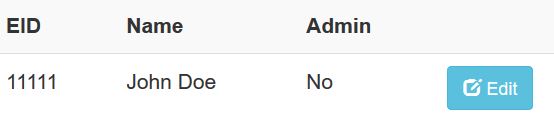
The supervisor is notified if the account has been successfully edited.

### Search Employee

Selecting the ‘Search Employee’ function allows a supervisor to look up the account information of a given employee.



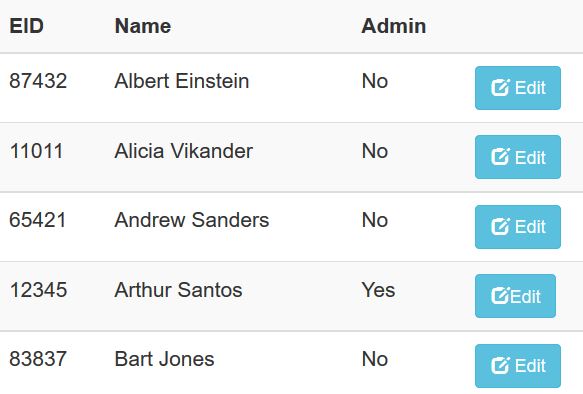
The supervisor is able to search for employee information based on Name, Employee ID, or Admin status. After inputting the appropriate search information, the desired employee’s information is displayed.



Optionally, the supervisor is also able to press the ‘Edit’ button and change the information for that specific employee.

### View Employees

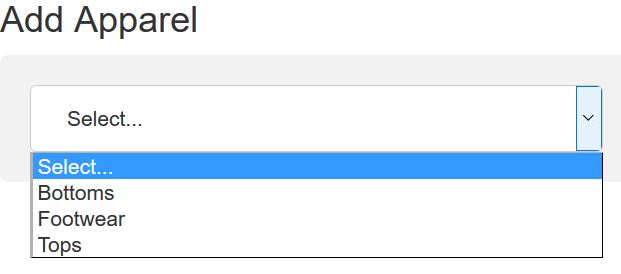
Selecting the ‘View Employees’ function allows a supervisor to view the account information of all employees.



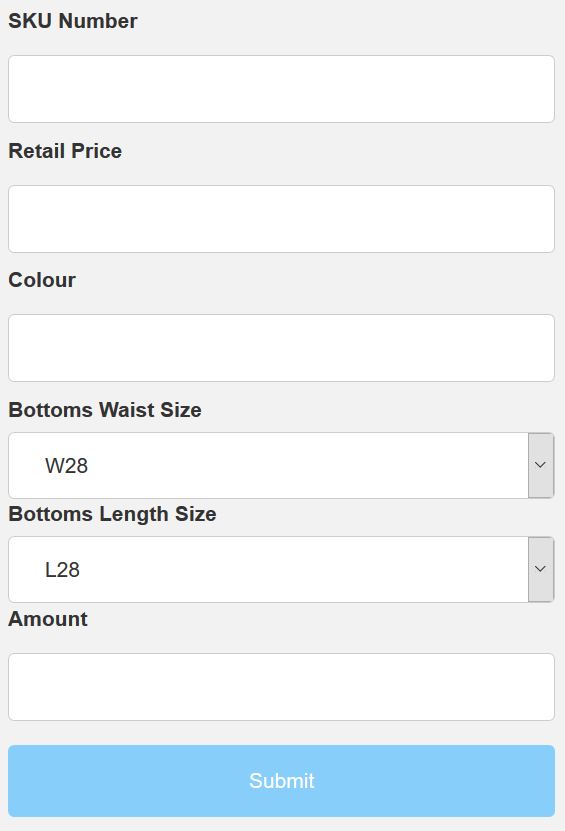
Optionally, the supervisor is also able to press the ‘Edit’ button and change the information for any employee listed.

### Add Apparel

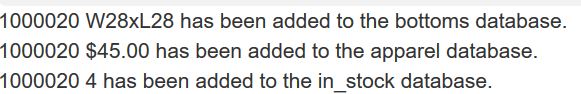
Selecting the ‘Add Apparel’ function allows a supervisor create a new item of apparel in the database.



After selecting what type of apparel they are adding, the supervisor is able to input information relevant to that item.

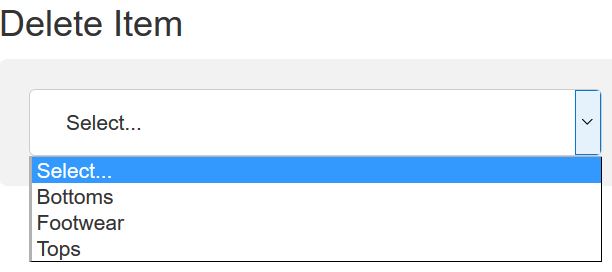


After filling out the necessary information, the supervisor is notified if the item has been successfully added.

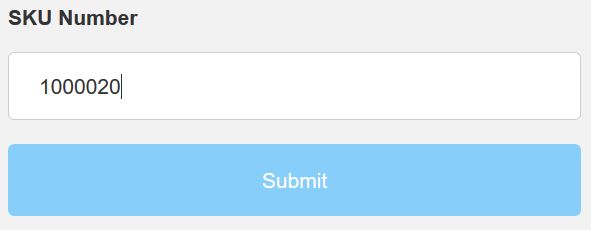


### Delete Apparel

Selecting the ‘Delete Apparel’ function allows a supervisor to remove an item of apparel from the database.



After selecting what type of apparel they are removing, the supervisor is able to input the SKU# of the item they want to remove.



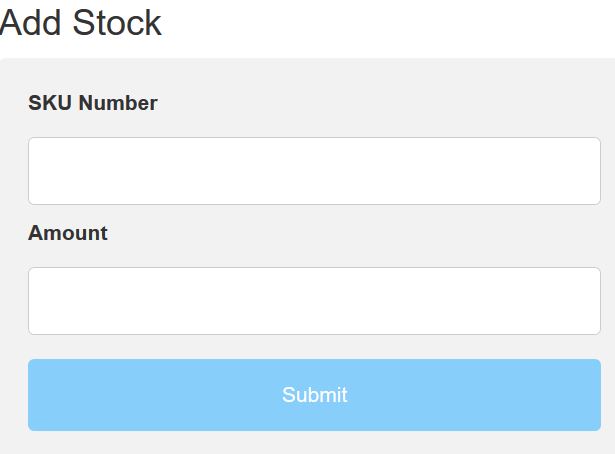
The supervisor is notified if the item has been successfully deleted.



*Note: Deleting an item from the database will not remove information from the ‘Stolen’ or ‘Write-Off’ tables.*

### Add Stock

Selecting the ‘Add Stock’ function allows a supervisor to add to the stock of an item that already exists in the inventory system.

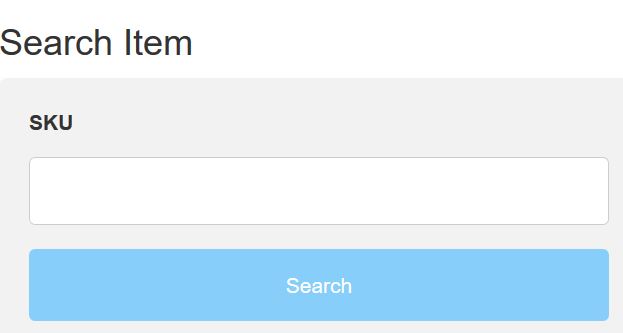


After inputting the desired item’s SKU# and the amount to add, the supervisor clicks the ‘Submit’ button to finish adding stock.

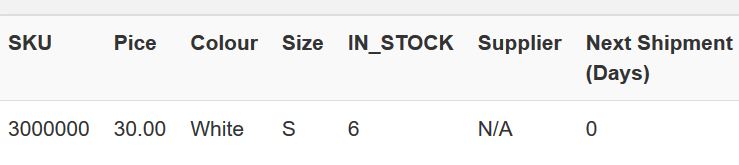


The supervisor is notified more stock has been successfully added.

### Search Apparel



The Search Apparel function allows a user to enter a SKU# for an item of apparel and receive information about that item.



A user is able to receive information about price, colour, size, and stock level of the specified piece of apparel. If there are no items currently in stock for that piece of apparel supplier information is displayed of which supplier carries the item as well as the estimated shipping time of receiving fresh stock.



### View Apparel

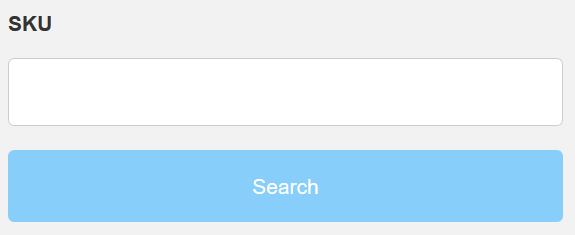
The View Apparel page allows a user to view all items of apparel currently in the database. A user will be able to view the price, colour, size, and number of items in stock.



All items are sorted by apparel type.

### Sale

The sale function allows a user to enter a SKU# for an item that is being purchased and have the inventory system adjust accordingly.

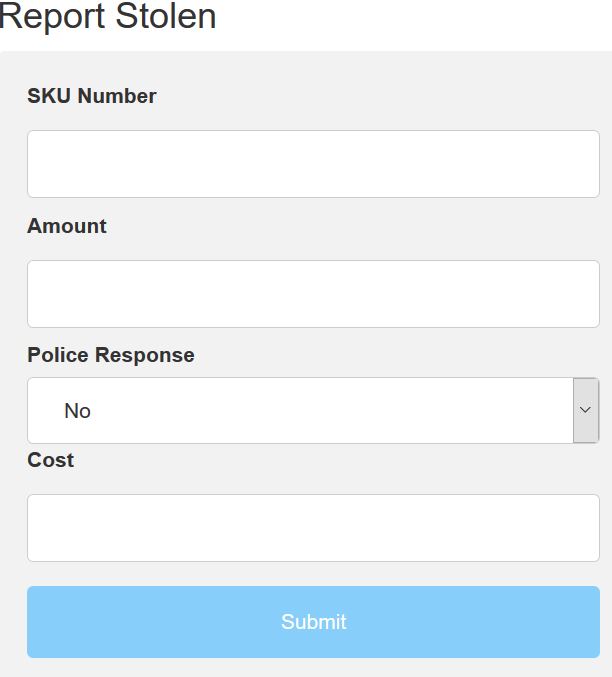


When the item has been purchased, the user is notified.



### Stolen

Selecting the ‘Stolen’ function allows a supervisor to log information about items that have been stolen.



After inputting the stolen item’s SKU#, number stolen, cost, and indicated whether there was a Police response, the supervisor clicks the ‘Submit’ button to log the stolen merchandise.



The supervisor is notified if the stolen item has been successfully logged.

### Write Off

Selecting the ‘Write Off’ function allows a supervisor to log information about items that have been written off.



After inputting the SKU# and amount of desired item to be written off, the supervisor clicks the ‘Submit’ button to log the written-off merchandise.



The supervisor is notified if the written-off item has been successfully logged.

### Logging Out

When a user is ready to log out of the inventory system, they will first need to click the Options menu button in the top-right of the window.



This brings up the options menu, which includes a logout function.



Logging out takes the user back to the login screen.

# Appendix B – Relationship Listing

**Function**: Check\_Position

**Inputs**: @EID @Password

**Output**:  AdminStatus

**Pseudocode**:

Connect to the database

Query =  SELECT  e.AdminStatus

               FROM  Employee as e

               WHERE e.EID  = @EID and e.Password = @Password

Parse Query

Execute Query

Close connection to database

**Function**: Sale

**Inputs**: @SKU#

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock - 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Return

**Inputs**: @SKU#

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock + 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Search/View Apparel

**Inputs**: @SKU#

**Output**:  Retail\_Price, #In\_Stock, Shipping\_Time

**Pseudocode**:

Connect to the database

Query =  SELECT  a.Retail\_Price, i.#In\_Stock, s.Shipping\_Time

               FROM  Apparel as a, In\_Stock as I, Supplier as s

               WHERE @SKU# = a.SKU# = i.SKU# = s.SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Add Apparel Footwear

**Inputs**: @SKU#, @Retail\_Price, @Colour, @Shoe\_Size, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Apparel

               VALUES (@SKU#, @Retail\_Price, @Colour)

Parse Query

Execute Query

Query =  INSERT INTO Footwear

               VALUES (@SKU#, @Shoe\_Size)

Parse Query

Execute Query

Query =  INSERT INTO In\_Stock

               VALUES (@SKU#, @#In\_Stock)

Parse Query

Execute Query

Close connection to database

**Function**: Add Apparel Outerwear

**Inputs**: @SKU#, @Retail\_Price, @Colour, @Size, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Apparel

               VALUES (@SKU#, @Retail\_Price, @Colour)

Parse Query

Execute Query

Query =  INSERT INTO Outerwear

               VALUES (@SKU#, @Size)

Parse Query

Execute Query

Query =  INSERT INTO In\_Stock

               VALUES (@SKU#, @#In\_Stock)

Parse Query

Execute Query

Close connection to database

**Function**: Add Apparel Tops

**Inputs**: @SKU#, @Retail\_Price, @Colour, @Size, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Apparel

               VALUES (@SKU#, @Retail\_Price, @Colour)

Parse Query

Execute Query

Query =  INSERT INTO Tops

               VALUES (@SKU#, @Size)

Parse Query

Execute Query

Query =  INSERT INTO In\_Stock

               VALUES (@SKU#, @#In\_Stock)

Parse Query

Execute Query

Close connection to database

**Function**: Add Apparel Bottoms

**Inputs**: @SKU#, @Retail\_Price, @Colour, @Waist\_Size, @Length, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Apparel

               VALUES (@SKU#, @Retail\_Price, @Colour)

Parse Query

Execute Query

Query =  INSERT INTO Bottoms

               VALUES (@SKU#, @Waist\_Size, @Length)

Parse Query

Execute Query

Query =  INSERT INTO In\_Stock

               VALUES (@SKU#, @#In\_Stock)

Parse Query

Execute Query

Close connection to database

**Function**: Delete Apparel

**Inputs**: @SKU#

**Output**:

**Pseudocode**:

Connect to the database

Query =  DELETE FROM Apparel, (Footwear or Outerwear or Tops or Bottoms), In\_Stock

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Edit Stock

**Inputs**: @SKU#, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = @#In\_Stock

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Add Stock

**Inputs**: @SKU#, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock + @#In\_Stock

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Remove Stock

**Inputs**: @SKU#, @#In\_Stock

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock - @#In\_Stock

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Update\_WO

**Inputs**: @SKU#

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock - 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Query =  UPDATE Write\_Offs

               SET  #WO = #WO + 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Update\_Stolen

**Inputs**: @SKU#

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  In\_Stock

               SET  #In\_Stock = #In\_Stock - 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Query =  UPDATE Stolen

               SET  #Stolen = #Stolen + 1

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**: Order

**Inputs**: @SKU#, @Shipping\_Time

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE Supplier

               SET  Shipping\_Time = @Shipping\_Time

               WHERE SKU# = @SKU#

Parse Query

Execute Query

Close connection to database

**Function**:New Order

**Inputs**: @SKU#,@Name, @Shipping\_Cost @Shipping\_Time

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Supplier

               VALUES  (@Shipping\_Time, @Name, @Shipping\_Cost, @Shipping\_Time)

Parse Query

Execute Query

Close connection to database

**Function**: Add Employee

**Inputs**: @EID, @FName, @LName @Password, @AdminStatus

**Output**:

**Pseudocode**:

Connect to the database

Query =  INSERT INTO Employee

               VALUES (@EID, @FName, @LName @Password, @AdminStatus)

Parse Query

Execute Query

Close connection to database

**Function**: Delete Employee

**Inputs**: @EID

**Output**:

**Pseudocode**:

Connect to the database

Query =  DELETE FROM Employee

               WHERE EID = @EID

Parse Query

Execute Query

Close connection to database

**Function**: Search/View Employee

**Inputs**: @EID

**Output**:   FName, LName, AdminStatus

**Pseudocode**:

Connect to the database

Query =  SELECT  FName, LName, AdminStatus

               FROM  Employee

               WHERE EID = @EID

Parse Query

Execute Query

Close connection to database

**Function**: Edit Employee

**Inputs**: @EID, @FName, @LName, @Password, @AdminStatus

**Output**:

**Pseudocode**:

Connect to the database

Query =  UPDATE  Employee

               SET FName = @FName, LName = @LName, Password = @Password,  AdminStatus = @AdminStatus

               WHERE EID = @EID

Parse Query

Execute Query

Close connection to database