

CPSC 471 Final Project Report

Daniel Jin 30107081

Minsu Kim 30068971

Steven Susanto 30073781

Group 38

# Table of content

[**Table of content 1**](#_e9daovttswol)

[**Abstract 3**](#_zemcwcdyn832)

[**Introduction 3**](#_aj78d4au7pbl)

[Address the problem 3](#_u9nqq9v972bg)

[**Project design 4**](#_tcid9ljdyqi1)

[Different users of the system 4](#_cgggzo88ikn7)

[ER diagram 5](#_h6rq5sjhny0y)

[Changes 6](#_1jumdheppcbt)

[**Implementation 7**](#_nvgwh2d21xz)

[Relational model diagram 7](#_81ksx8l86f8v)

[Changes 8](#_hsix77to2fdk)

[DBMS 8](#_mhgtx9hmfj4)

[Create table statements 8](#_odkjhbyyj5rl)

[Select, insert, update, delete statements 11](#_50yy3qp7ytk5)

[Sign in 11](#_vt2f5ajalq2d)

[Add employee 11](#_23ogd5uggkbk)

[Employees that are not supervisors 11](#_fw5f2s5p42bo)

[Employees under supervision by certain supervisor 11](#_aiptbfeweu52)

[Add/update food 12](#_hmufx9sle0vp)

[Add/update clothe 12](#_ylbg7bgstgk3)

[Assign/complete/pick up orders for back employee 12](#_ama20o5zl6cg)

[Preview order 12](#_lejtozy15h8t)

[Change password 13](#_giybi0b2rwd8)

[Change role/supervisor 13](#_dpmbisn56ahw)

[Show unassigned/ready for pick up/incomplete/completed orders 13](#_x5snyounpefj)

[Create clothing order 13](#_wert2p6wrb79)

[Create food order 14](#_3tj6r24279h6)

[List of clothes/foods 14](#_b5c5zmbgn29)

[ReplenishC/F 15](#_lmv7hxfib078)

[**Visual interface 15**](#_wiggzm4sdio5)

[**User manual 16**](#_r6qt6s36t7pm)

[Sign in page 16](#_jdneo84gz6h2)

[Home pages 16](#_pcymop2h9tet)

[Supervisor 16](#_gvl4d599uf7y)

[Employees page 17](#_sd4y1sfa29xp)

[Changing employee role 17](#_s0clsofn9pxq)

[Changing employee’s supervisor 18](#_6gx9jx3qca5o)

[Add new employee 18](#_cupjk6wi0on)

[Replenish food inventory page 19](#_uhbfvgz0u7ea)

[Update food quantity 19](#_3hxha8gaioyw)

[Adding new food 20](#_tbj9k7zft0yb)

[Replenish clothing inventory page 20](#_klleb1xxgcgz)

[Update clothing quantity 21](#_mt69kgxu96tv)

[Add new clothing 21](#_4h0isp7cpvfs)

[Front Employee 22](#_iv5v07rlkdq9)

[Start a food order page 23](#_j9d6ppzcfwnk)

[Start a clothing order page 24](#_y7k5682yk728)

[View fulfilled orders page 25](#_kmqzc6c0gtcm)

[Back Employee 26](#_5pgrr4m84ca5)

[Unassigned orders page 26](#_hc5ju7r9jjqc)

[Orders assigned to me page 27](#_img771rc3723)

[Orders ready for pick up page 28](#_r3dfis990dr)

[Orders completed and picked up page 29](#_3kjri0ls9ga1)

[Profile page 30](#_qehd6jxuyvic)

[Supervisor profile page 31](#_kdf0vutnoou7)

[Front/Back employee profile page 31](#_m9ca0hq3u1ld)

[Change password page 32](#_z3rx8nk2c2xn)

# Abstract

The Calgary Food/ Clothing Bank is an organisation that collects and distributes food or clothing to individuals in need. However, they currently use a manual, pen and paper system to receive and send out donations. This is an extremely inefficient process that is also prone to human errors. We believe that the system can easily be improved using a database management system that the organisation can benefit from. The objective of our project is to put together a database system connected to a web application using the knowledge we learned from CPSC 471.

# Introduction

The Calgary Food/Clothing Bank currently uses a manual system to receive and send out donations to folks in need. In any storage system, writing down which item goes in and out of the inventory and then manually tallying is such a tedious and time consuming job. In addition, there is also human error to be taken into consideration. For instance, the employee runs out of pen ink causing customers to wait or losing the paper where they have written down orders resulting in inventory data not adding up. Especially on a busy day, where the employees need to be quick to accommodate other clients, they could miss a few things.

To overcome this challenge, our project aims to create a database system connected to a front end user interface web application for a friendly user experience to help solve this problem for The Calgary Food/Clothing Bank. We started by creating an Enhanced Entity Relationship Diagram (EERD), which is then converted into a relational model. The relational models are then translated into UML diagrams as well as planning out the sequence diagram which both are used as the base for our final system.

## Address the problem

Our system is built specifically for the employees working in The Calgary Food/Clothing Bank. Through the frontend user interface, employees are able to check and update existing items in the inventory, keep track of orders made by clients as well as the order’s progress. Ultimately, this is to increase efficiency of the entire ordering process as well as reduce the chance of human errors by a significant amount.

# Project design

## Different users of the system

There are a total of three different users for our system, supervisor, front employee that deals with the clients order and the back employee that works on packaging the orders. Each role is unique and their job is as follows

* Supervisor: The supervisor’s job is to replenish food or clothing stock that they are responsible for. This means that ,once in a while, the supervisor will update the database inventory to match the physical inventory. For example, if food were to go bad or clothes were to go missing, the inventory will get updated accordingly. Another example is if a new shipment of food or clothing arrived, the inventory will get updated accordingly. Supervisors can also add new foods or clothes to the database with a given quantity (duplicate items will not get added). The supervisor that adds an item will be the one in charge of replenishing/updating that item.

Supervisors can also add new employees into the database (with unique usernames), change the roles of existing employees (that are not supervisors), and change the supervisor for existing employees.

Note that:

All newly created accounts get a default password of 123, which can later be changed by the respective user.

When a supervisor adds an employee that is not a supervisor, the supervisor for that new employee will be the one who created the employee. If the supervisor adds a supervisor employee, the supervisor for the new employee will be no one. If the supervisor changes a non-supervisor employee to a supervisor, the supervisor of the new supervisor will be no one. If the supervisor changes the role of a back employee to a front employee (or vice versa) the supervisor remains the same.

* Front Employee: The front employee’s job is to take orders from the clients. In our start order page, the front employee can edit the number of family members based on adults and kids. Then, based on the type of the order, they can select food for the family if the inventory exists. Note that, the system uses the same logic for clothes orders, only the item descriptions would be different. They will also be given a total of recommended amount of calories per food order or a recommended number of items per clothing order. They have a choice to either go under or over the recommended number. After confirming an order, the order will get set to a list of orders that a back employee can choose to take. Before confirming an order, the front employee has a chance to preview the order which they can then edit if they need to. Front employees can also view all the orders that they have confirmed.
* Back Employee: The back employee’s job is to package the orders created by the front employee. Our system helps keep track of which orders are unassigned, incomplete, completed as well as picked up by the client.
  + *Unassigned orders:* These are orders that are not assigned to a back employee yet. Thus, the back employee chooses the orders he wants to assign themselves.
  + *Incomplete orders*: These are orders that are assigned to a back employee but are still in the process of packing the order.
  + *Completed orders*: These are orders that are fully packaged by the back employee and ready for pick up.
  + *Completed and picked up orders*: These are orders that have been completed and picked up by a client.

## ER diagram

### 

### 

### Changes

The ER diagram for the project is represented by the figure above (also added at the end of the documentation for clarity). There were only minor changes made to the design from our initial design, such as: adding simple attributes to our already existing entities and removing simple attributes.

The things we added:

* Username and password attributes to the “Employee” entity. This change was made so that we can authenticate the users that sign into our website. This was also to address the feedback that was given after handing in our project functional model.
* Role attribute was also added to the “Employee” entity. This simply states what role the employee has (supervisor, back employee, or front employee).
* Ready\_for\_pick\_up attribute was added to the “Order” entity. If the picked\_up attribute is used to determine if an order was picked up or not by a family, the ready\_for\_pick\_up attribute is used to determine if an order was completely packed by a back employee (ready for pick up or not).
* Type attribute was also added to the “Order” entity. This simply states what type of order the order is (food or clothes).
* Description attribute was added to the “Clothes” entity. This is used to describe the item. For some cases, an employee might not know what the clothing is without reading the description.
* Changed the attribute name “Cli\_id” to “Fam\_id” for the “Family” entity. This is so that it is more legible, and removes any confusion.
* Changed the key for entities “Child” and “Adult” from “Name” to “id”. This was changed because names might not be unique. Also, we do not need to know the name since the authentication of a client is not handled by our system but rather with the other system provided by the Calgary Food/Clothing bank that has access to personal information such as income, SIN, etc.

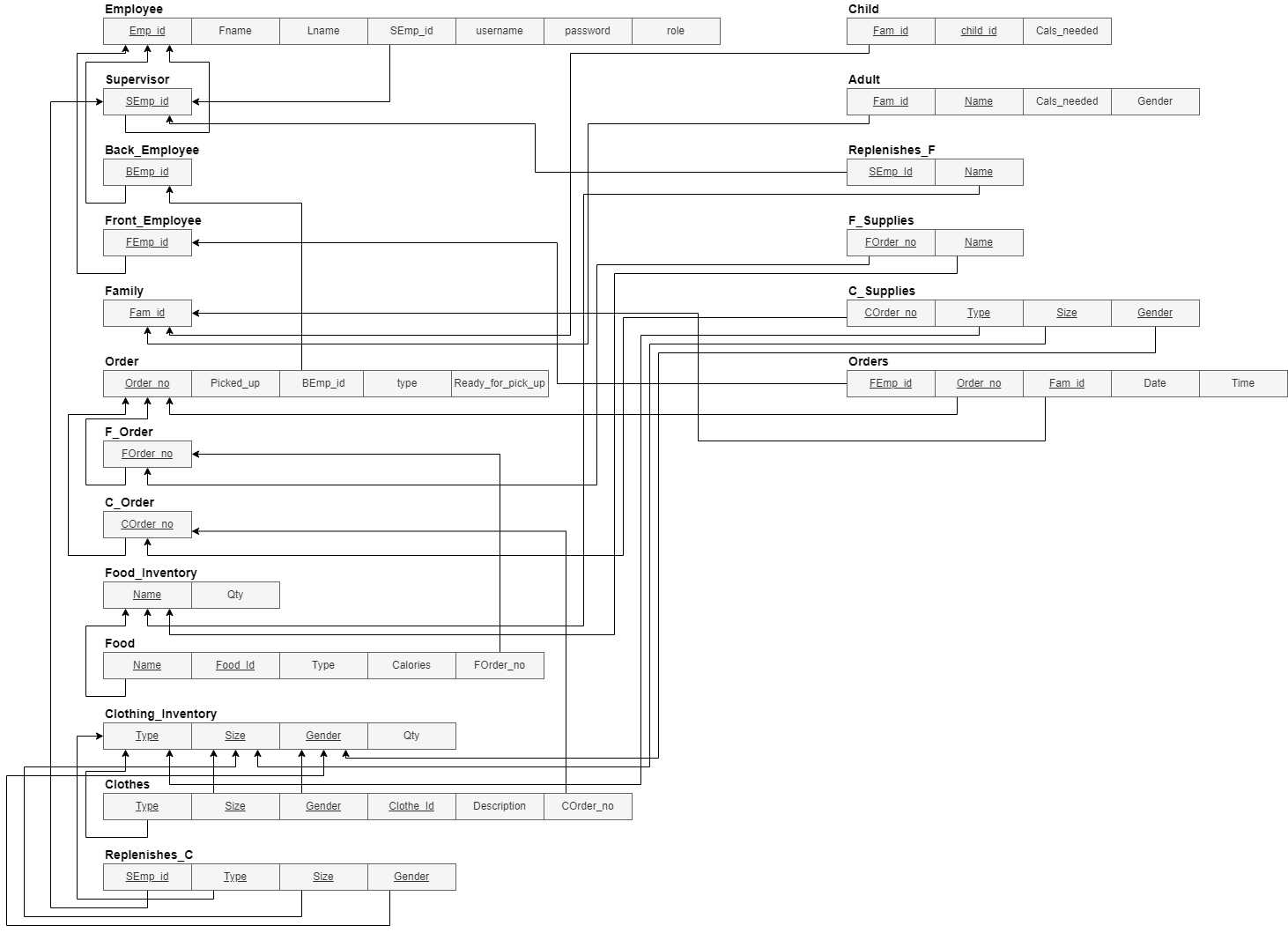
The things we removed:

* Age attribute from the entities “Child” and “Adult”. Since the calories needed for an age group varies a lot we instead just divide the members of the family into two age groups, adult or child. Therefore, we did not need to know the age of the clients.
* Gender attribute from the entity “Child”. Since the required amount of calories for children does not change whether the child is male or female, we got rid of the gender attribute.
* Total\_cals\_needed derived attribute from the “Family” entity, we instead calculate this value in our code.
* Condition and Brand attributes from the “Clothes” entity. In a charity food bank, the condition will always be good enough for people to use and the brand should not matter. Therefore, we got rid of these two attributes.
* Description attribute from the “Food” entity. We assume that employees will know what type of foods they carry and don’t require descriptions.

Our EER diagram for The Calgary Food/Clothing Bank was created based on our knowledge of the organisation. Having had the chance to volunteer and opportunities to speak with the employees gave us an insight on how the organisation functions currently. The important components of the ordering system include the employees and its specialisations (supervisor, front employee and back employee), clients or families, the order itself as well as the inventory of food and clothing.

# Implementation

## Relational model diagram



### 

### Changes

The relational model diagram for the project is represented by the figure above (also added at the end of the documentation for clarity). The changes that were made to the ER diagram described above is also reflected in the relational model diagram and how we decided to implement the system.

## DBMS

For the project we use MariaDB as our database management system. Note that MariaDB is a fork from MySQL and is compatible with prior versions of MySQL.

## Create table statements

To create the tables in our database these are the SQL statements:

CREATE TABLE `Supervisor` (

`Semp\_id` int(10) REFERENCES Employee(`Emp\_id`),

PRIMARY KEY (`Semp\_id`));

CREATE TABLE `Employee` (

`Emp\_id` int(10) NOT NULL AUTO\_INCREMENT,

`Fname` varchar(30) NOT NULL,

`Lname` varchar(30) NOT NULL,

`Semp\_id` int(10),

`username` varchar(30) NOT NULL UNIQUE,

`password` varchar(30) NOT NULL,

`role` varchar(15) NOT NULL check (`role` in ('Supervisor', 'Front', 'Back')),

PRIMARY KEY (`Emp\_id`),

FOREIGN KEY (`Semp\_id`) REFERENCES Supervisor(`Semp\_id`));

CREATE TABLE `Back\_Employee` (

`Bemp\_id` int(10) REFERENCES Employee(`Emp\_id`),

PRIMARY KEY (`Bemp\_id`));

CREATE TABLE `Front\_Employee` (

`Femp\_id` int(10) REFERENCES Employee(`Emp\_id`),

PRIMARY KEY (`Femp\_id`));

CREATE TABLE `Family` (

`Fam\_id` int(10) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`Fam\_id`));

CREATE TABLE `Order` (

`Order\_no` int(10) NOT NULL AUTO\_INCREMENT,

`Picked\_up` bit(1) NOT NULL,

`Bemp\_id` int(10),

`type` varchar(15) NOT NULL check (`type` in ('Food', 'Clothe')),

`Ready\_for\_pick\_up` bit(1) NOT NULL,

PRIMARY KEY (`Order\_no`),

FOREIGN KEY (`Bemp\_id`) REFERENCES Back\_Employee(`Bemp\_id`));

CREATE TABLE `F\_Order` (

`Forder\_no` int(10) NOT NULL REFERENCES `Order`(`Order\_no`),

PRIMARY KEY (`Forder\_no`));

CREATE TABLE `C\_Order` (

`Corder\_no` int(10) NOT NULL REFERENCES `Order`(`Order\_no`),

PRIMARY KEY (`Corder\_no`));

CREATE TABLE `Food\_Inventory` (

`name` varchar(100) NOT NULL,

`qty` int(10) unsigned NOT NULL,

PRIMARY KEY (`name`));

CREATE TABLE `Food` (

`Food\_id` varchar(40) NOT NULL UNIQUE,

`name` varchar(100) NOT NULL REFERENCES `food\_inventory`(`name`),

`type` varchar(10) NOT NULL check (`type` in ('FV', 'Grain', 'Meat', 'Dairy', 'Other')),

`calories` int unsigned NOT NULL,

`Forder\_no` int(10),

PRIMARY KEY (`name`, `Food\_id`),

FOREIGN KEY (`Forder\_no`) REFERENCES `F\_Order`(`Forder\_no`));

CREATE TABLE `Clothing\_Inventory` (

`type` varchar(100) NOT NULL,

`size` varchar(5) check (`size` in ('XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL')),

`gender` char(1) check (`gender` in ('M', 'F', 'U')),

`qty` int(10) unsigned NOT NULL,

PRIMARY KEY (`type`, `size`, `gender`));

CREATE TABLE `Clothe` (

`Clothe\_id` varchar(40) NOT NULL UNIQUE,

`type` varchar(100) NOT NULL REFERENCES `clothing\_inventory`(`type`),

`size` varchar(5) check (`size` in ('XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL')) REFERENCES `clothing\_inventory`(`size`),

`gender` char(1) check (`gender` in ('M', 'F', 'U')) REFERENCES `clothing\_inventory`(`gender`),

`description` varchar(100) NOT NULL,

`Corder\_no` int(10),

PRIMARY KEY (`type`, `size`, `gender`,`Clothe\_id`),

FOREIGN KEY (`Corder\_no`) REFERENCES `C\_order`(`Corder\_no`));

CREATE TABLE `Replenish\_C` (

`Semp\_id` int(10) REFERENCES Supervisor(`Semp\_id`),

`type` varchar(100) NOT NULL REFERENCES clothing\_inventory(`type`),

`size` varchar(5) check (`size` in ('XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL')) REFERENCES clothing\_inventory(`size`),

`gender` char(1) check (`gender` in ('M', 'F', 'U')) REFERENCES clothing\_inventory(`gender`),

PRIMARY KEY (`Semp\_id`, `type`, `size`, `gender`));

CREATE TABLE `Child` (

`Fam\_id` int(10) NOT NULL REFERENCES `family`(`Fam\_id`),

`child\_id` varchar(40) NOT NULL UNIQUE,

`cals\_needed` int(10) NOT NULL,

PRIMARY KEY (`Fam\_id`, `child\_id`));

CREATE TABLE `Adult` (

`Fam\_id` int(10) NOT NULL REFERENCES `family`(`Fam\_id`),

`adult\_id` varchar(40) NOT NULL UNIQUE,

`cals\_needed` int(10) NOT NULL,

`gender` char(1) NOT NULL check (`gender` in ('M', 'F')),

PRIMARY KEY (`Fam\_id`, `adult\_id`));

CREATE TABLE `Replenish\_F` (

`Semp\_id` int(10) REFERENCES Supervisor(`Semp\_id`),

`name` varchar(100) REFERENCES food\_inventory(`name`),

PRIMARY KEY (`Semp\_id`, `name`));

CREATE TABLE `F\_Supplies` (

`Forder\_no` int(10) NOT NULL REFERENCES `F\_Order`(`Forder\_no`),

`name` varchar(100) NOT NULL REFERENCES `food\_inventory`(`name`),

PRIMARY KEY (`Forder\_no`,`name`));

CREATE TABLE `C\_Supplies` (

`Corder\_no` int(10) NOT NULL REFERENCES `C\_order`(`Corder\_no`),

`type` varchar(100) NOT NULL REFERENCES `clothing\_inventory`(`type`),

`size` varchar(5) check (`size` in ('XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL')) REFERENCES `clothing\_inventory`(`size`),

`gender` char(1) check (`gender` in ('M', 'F', 'U')) REFERENCES clothing\_inventory(`gender`),

PRIMARY KEY (`Corder\_no`,`type`,`size`,`gender`));

CREATE TABLE `Orders` (

`Femp\_id` int(10) NOT NULL REFERENCES `Front\_Employee`(`Femp\_id`),

`Order\_no` int(10) NOT NULL REFERENCES `order`(`Order\_no`),

`Fam\_id` int(10) NOT NULL REFERENCES `family`(`Fam\_id`),

`date` varchar(20) NOT NULL,

`time` varchar(20) NOT NULL,

PRIMARY KEY (`Femp\_id`, `Order\_no`, `Fam\_id`));

## 

## Select, insert, update, delete statements

### Sign in

* Select \* from `employee` where username='$username' and password='$password'

### Add employee

* insert into `employee` (Fname, Lname, Semp\_id, username, password, role) values ('$fname', '$lname', NULL, '$user', '$pw', '$role')
* insert into `back\_employee` (Bemp\_id) values ('$bemp\_id')
* insert into `front\_employee` (Femp\_id) values ('$femp\_id')
* Select \* from `employee` where Emp\_id = '$id'

### Employees that are not supervisors

* Select \* from `employee` where role !='Supervisor'

### Employees under supervision by certain supervisor

* Select \* from `employee` where role !='Supervisor' and Semp\_id = '$id'

### Add/update food

* Select \* from `food\_inventory` where name='$name'
* insert into `food\_inventory` (name, qty) values ('$name', '$qty')
* insert into `replenish\_f` (Semp\_id, name) values ('$id', '$name')
* insert into `food` (Food\_id, name, type, calories) values ('$f\_id','$name', '$type', '$calories')
* select COUNT(name) as qty from `food` where name='$name' and Forder\_no IS NULL
* select COUNT(name) as qty from `food` where name='$name'
* update `food\_inventory` set qty='$count' where name='$name'
* delete from `food` where name='$name' limit $update\_qty
* delete from `food` where name='$name' and Forder\_no IS NULL limit $update\_qty
* delete from `replenish\_f` where name='$name'
* delete from `food\_inventory` where name='$name'

### Add/update clothe

* Select \* from `clothing\_inventory` where type='$type' and size='$size' and gender='$gender'
* insert into `clothing\_inventory` (type, size, gender, qty) values ('$type', '$size', '$gender', '$qty')
* insert into `replenish\_c` (Semp\_id, type, size, gender) values ('$id', '$type', '$size', '$gender')
* insert into `clothe` (Clothe\_id, type, size, gender, description) values ('$c\_id','$type', '$size', '$gender', '$desc')
* select COUNT(Clothe\_id) as qty from `clothe` where type='$type' and size='$size' and gender = '$gender' and Corder\_no IS NULL
* update `clothing\_inventory` set qty='$count' where type='$type' and size='$size' and gender = '$gender'
* select COUNT(Clothe\_id) as qty from `clothe` where type='$type' and size='$size' and gender = '$gender'
* delete from `clothe` where type='$type' and size='$size' and gender = '$gender' limit $update\_qty
* delete from `replenish\_c` where type='$type' and size='$size' and gender = '$gender'
* delete from `clothing\_inventory` where type='$type' and size='$size' and gender = '$gender'

### Assign/complete/pick up orders for back employee

* update `order` set Bemp\_id=$user where Order\_no = $ordNo
* update `order` set Ready\_for\_pick\_up = 1 where Order\_no = $ordNo
* update `order` set picked\_up = 1 where Order\_no = $ordNo

### Preview order

* Select \* from `order` where Order\_no = '$order\_no'
* Select count(fam\_id) as x from `child` where `Fam\_id`='$fam\_id'
* Select count(fam\_id) as x from `adult` where `Fam\_id`='$fam\_id' and `gender`='M'
* Select count(fam\_id) as x from `adult` where `Fam\_id`='$fam\_id' and `gender`='F'
* Select \* from `food` where `Forder\_no`='$order\_no'
* Select COUNT(clothe\_id) as qty from `clothe` where `Corder\_no`='$order\_no'
* Select \* from `employee` where `Emp\_id`='$femp\_id'
* Select \* from `employee` where `Emp\_id`='$bemp\_id'
* Select c.type, c.size, c.gender ,COUNT(c.clothe\_id) from `clothe` as c where c.corder\_no = $order\_no GROUP BY c.type,c.size, c.gender
* Select f.name, f.type, f.calories ,COUNT(f.name) from `food` as f where f.Forder\_no = $order\_no GROUP BY f.name

### Change password

* update `employee` set password='$new\_pw' where Emp\_id = '$id'

### Change role/supervisor

* update `employee` set Semp\_id = NULL where Emp\_id='$id'
* update `employee` set role='$role1' where Emp\_id='$id'
* delete from `back\_employee` where Bemp\_id='$id'
* delete from `front\_employee` where Femp\_id='$id'
* update `employee` set Semp\_id='$super' where Emp\_id='$id'
* Select \* from `employee` where Emp\_id='$super'

### Show unassigned/ready for pick up/incomplete/completed orders

* Select \* From `Order` as o, `Orders` as os WHERE o.picked\_up = 1 and os.order\_no = o.order\_no
* Select \* From `Order` as O, `orders` as os WHERE O.Bemp\_id is NULL and os.order\_no = o.order\_no
* Select \* From `Order` as o, `Orders` as os WHERE o.picked\_up = 0 and o.Bemp\_id='$user' and o.Ready\_for\_pick\_up = 1 and os.order\_no = o.order\_no
* Select \* From `Order` as o, `Orders`as os WHERE o.picked\_up = 0 and o.Bemp\_id='$user' and o.Ready\_for\_pick\_up = 0 and os.order\_no = o.order\_no

### Create clothing order

* insert into `family` (`Fam\_id`) VALUES (NULL)
* insert into `adult` (Fam\_id, adult\_id, cals\_needed, gender) VALUES ('$fam\_id', '$member\_id', '$cal', 'M')
* insert into `adult` (Fam\_id, adult\_id, cals\_needed, gender) VALUES ('$fam\_id', '$member\_id', '$cal', 'F')
* insert into `child` (Fam\_id, child\_id, cals\_needed) VALUES ('$fam\_id', '$member\_id', '$cal')
* insert into `Order` (Picked\_up, Bemp\_id, type, Ready\_for\_pick\_up) values (0, NULL, 'Clothe', 0)
* insert into `C\_Order`(Corder\_no) values ('$order\_id')
* insert into `Orders` (Femp\_id, Order\_no, Fam\_id, date, time) values ('$id', '$order\_id', '$fam\_id', '$date', '$time')
* insert into `C\_supplies` (Corder\_no, type, size, gender) values ('$order\_id', '$type', '$size', '$gender')
* select \* from `clothe` where type='$type' and size='$size' and gender='$gender' and Corder\_no IS NULL
* update `clothe` set Corder\_no='$order\_id' where Clothe\_id='$clothe\_id'
* select COUNT(type) as qty from `clothe` where type='$type' and size='$size' and gender='$gender' and Corder\_no IS NULL
* update `clothing\_inventory` set qty='$count' where type='$type' and size='$size' and gender='$gender'

### Create food order

* insert into `family` (`Fam\_id`) VALUES (NULL)
* insert into `adult` (Fam\_id, adult\_id, cals\_needed, gender) VALUES ('$fam\_id', '$member\_id', '$cal', 'M')
* insert into `adult` (Fam\_id, adult\_id, cals\_needed, gender) VALUES ('$fam\_id', '$member\_id', '$cal', 'F')
* insert into `child` (Fam\_id, child\_id, cals\_needed) VALUES ('$fam\_id', '$member\_id', '$cal')
* insert into `Order` (Picked\_up, Bemp\_id, type, Ready\_for\_pick\_up) values (0, NULL, 'Food', 0)
* insert into `F\_Order`(Forder\_no) values ('$order\_id')
* insert into `Orders` (Femp\_id, Order\_no, Fam\_id, date, time) values ('$id', '$order\_id', '$fam\_id', '$date', '$time')
* insert into `F\_supplies` (Forder\_no, name) values ('$order\_id', '$name')
* select \* from `food` where name='$name' and Forder\_no IS NULL
* update `food` set Forder\_no='$order\_id' where Food\_id='$food\_id'
* select COUNT(name) as qty from `food` where name='$name' and Forder\_no IS NULL
* update `food\_inventory` set qty='$count' where name='$name'

### List of clothes/foods

* Select distinct fi.type, fi.size, fi.gender, fi.qty, f.description from `clothe` as f, `clothing\_inventory` as fi where f.type = fi.type and f.type = '$type'
* Select distinct fi.name, fi.qty, f.calories from `food` as f, `food\_inventory` as fi where f.name = fi.name and f.type = '$type'

### ReplenishC/F

* Select distinct ci.type, ci.size, ci.gender, ci.qty, c.description from `employee` as e, `clothing\_inventory` as ci, `clothe` as c, `replenish\_c` as r where e.Emp\_id='$user' and e.Emp\_id = r.Semp\_id and r.type = ci.type and r.size = ci.size and r.gender = ci.gender and ci.type = c.type and ci.size = c.size and ci.gender = c.gender
* Select distinct fi.name, fi.qty, f.type, f.calories from `employee` as e, `food\_inventory` as fi, `food` as f, `replenish\_f` as r where e.Emp\_id='$user' and e.Emp\_id = r.Semp\_id and r.name = fi.name and fi.name = f.name

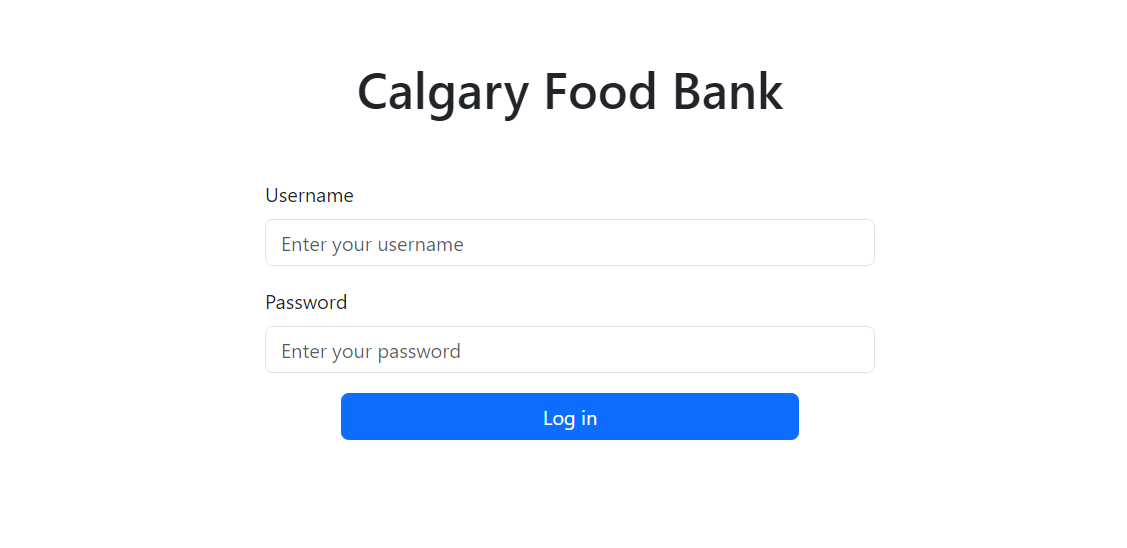
# Visual interface

We used the bootstrap library to create a clean, simple, cohesive look for our application. We made each clickable item medium to large size in order to allow easy clicking. For important buttons that initialise an action such as sign in button, profile button, add food button, add clothing button, etc. we decided to make the colour blue in order for it to stand out to our users. For buttons such as cancel order button, logout button, remove button, etc. we made the colours red to alert the users that the button might cause an action that is not wanted. The add button is coloured green to represent an addition function. Important words or numbers are bolded.

We also centre all our pages for uniformity and consistency. Data is given in a table format so that the users can easily consume the information. Additionally, the user interface has a minimum amount of text fields so that the user can click away which allows fast operations to be executed.

# User manual

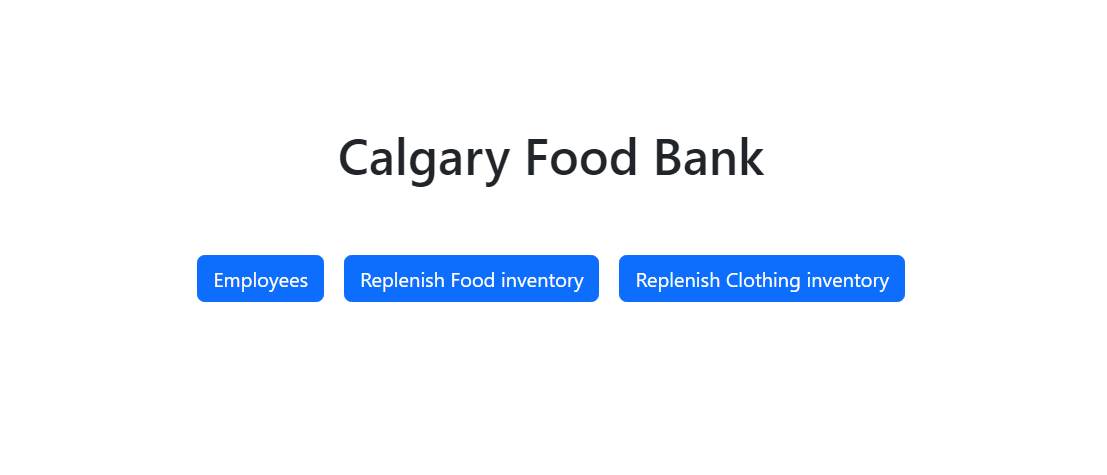
## Sign in page



The sign in page is used for all users regardless of the role and to sign in, users will input their username and password and if it matches, they will be taken to their home page, which is different for each role. If the credentials do not match, an error will be alerted.

## Home pages

### Supervisor

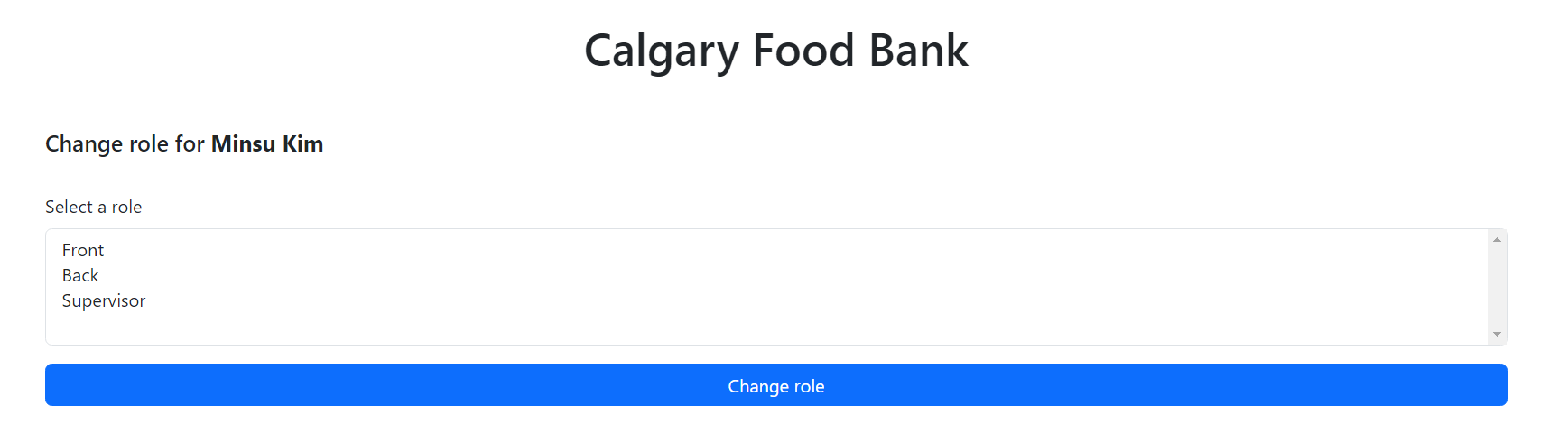


Home page for supervisor consists of three buttons Employees, Replenish Food inventory, and Replenish Clothing inventory:

#### Employees page

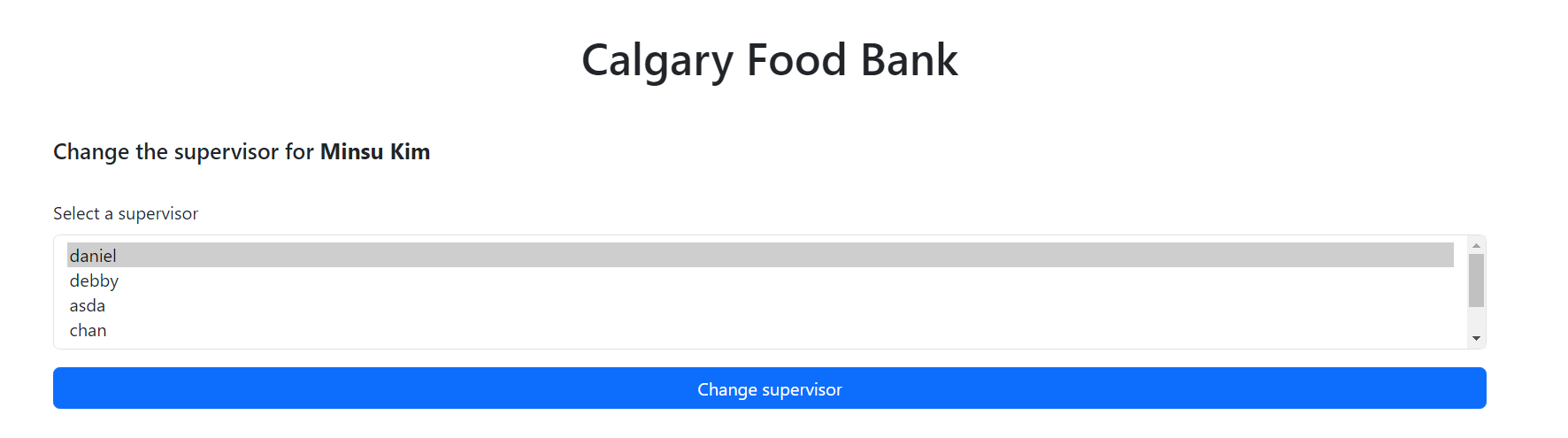
Employees section allows the supervisor to see the lists of employees in the database that are not a supervisor, and has the option of changing each employee’s roles and the supervisor that is in charge of that specific employee. Also, supervisors can add new employees, which is essentially a sign up page, where supervisors add new employees into the database and assign them username, password, and roles.

#### Changing employee role

****

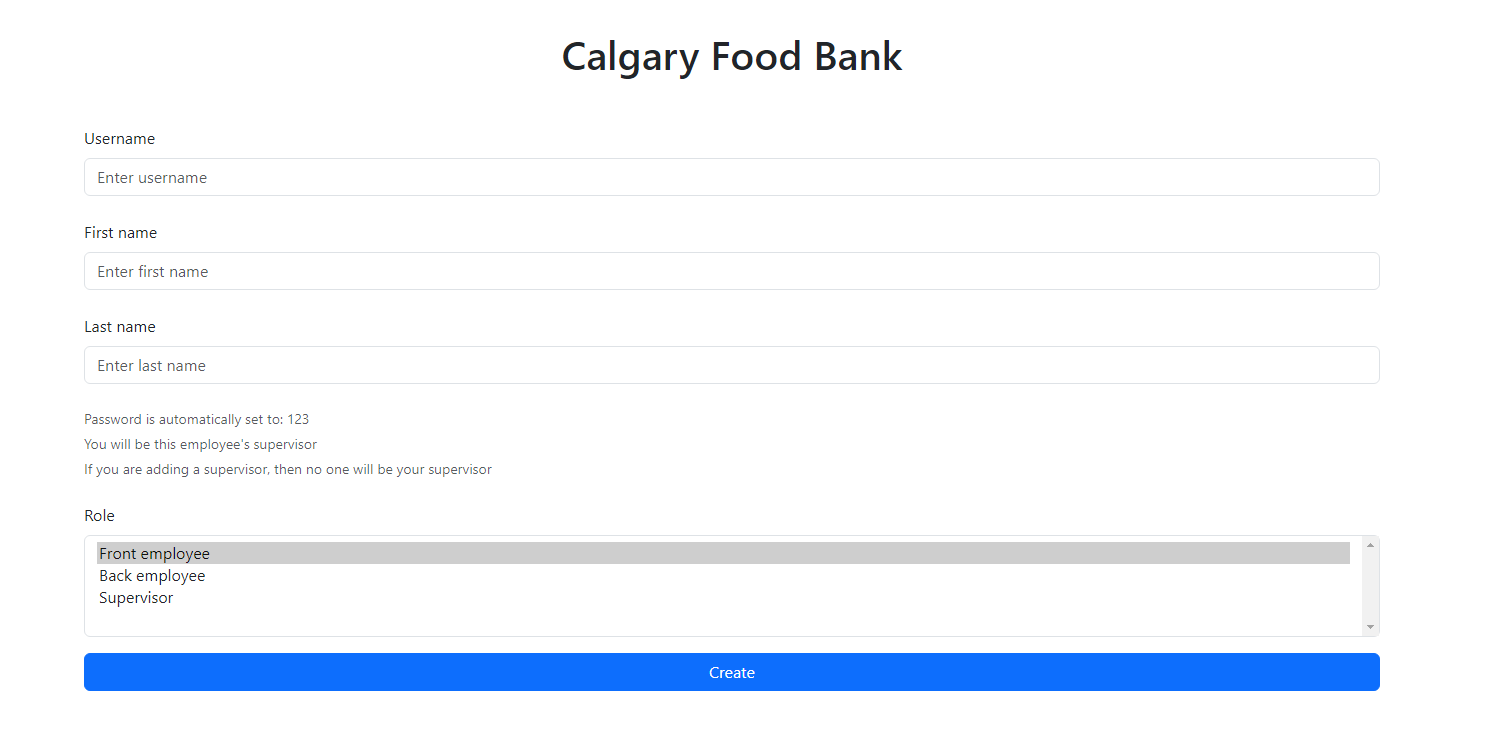
If the supervisor presses the “role” button from the employees page, then for that specific employee, the supervisor can assign a different role for the employee and it is done by selecting one of the three roles, and pressing change role. If the supervisor does not select any one of the roles and presses change role, an error message will pop up, and does not change anything in the database. If the role of a specific employee is changed to a supervisor, then the employee will be taken out of the list of the employees shown in the previous page. If the role change goes through successfully, then a message will pop up notifying the user that the role of the specific employee has changed to the new role. If an employee’s role is changed to a supervisor, its supervisor will now be no one.

#### Changing employee’s supervisor

****

If the supervisor presses the “supervisor” button from the employees page, then for that specific employee, the supervisor can assign a different supervisor. There will be a list of current supervisors from the database, and the supervisor can select any one of the supervisors and once the change supervisor button is pressed, then the employee’s supervisor will be updated to such. If the supervisor selects no supervisor, then an error message will pop up and it will not make any changes.

#### Add new employee



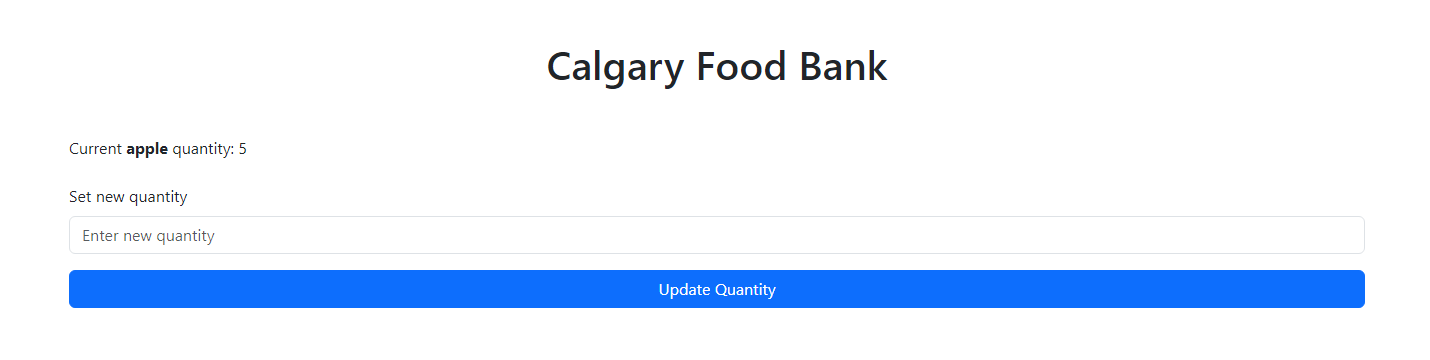
Pressing the “Add new employee” button from the “Employee” page will take the user to a new page, where they submit a form to add a new employee to the database, and they input their name, username, and the password will be set to a default password. The supervisor also sets the role of the newly added employee. Finally, the supervisor of the new employee (if not a supervisor) will be the user that created the employee. Usernames are unique, therefore if there already exists someone in the database with the same username an error will be alerted.

#### Replenish food inventory page



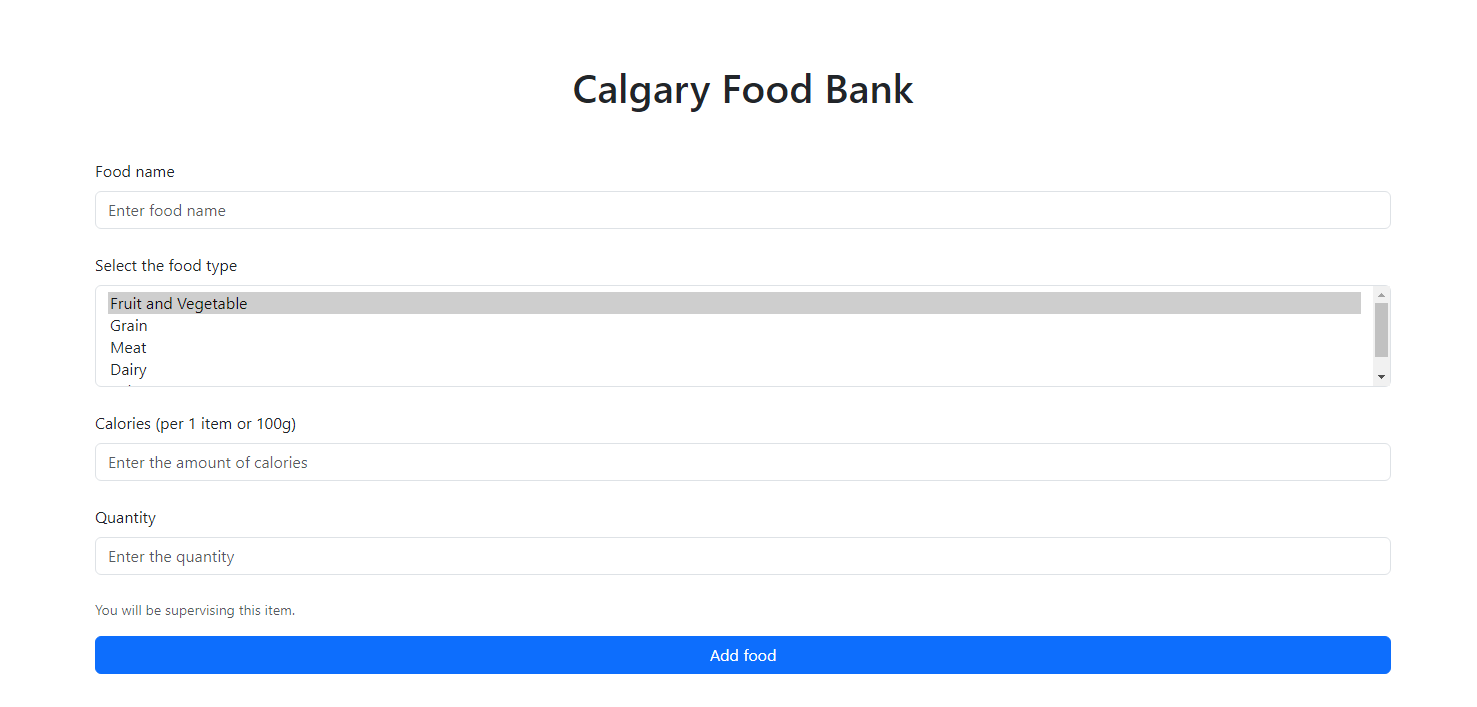
Replenish food inventory page shows a list of foods that are registered in the database currently, and allows the supervisor to update the quantity of each food. They can either increase the quantity or decrease it. It also has an add food button, which allows the supervisor to register a new food into the database, and it will be updated in the list once it is added.

#### Update food quantity



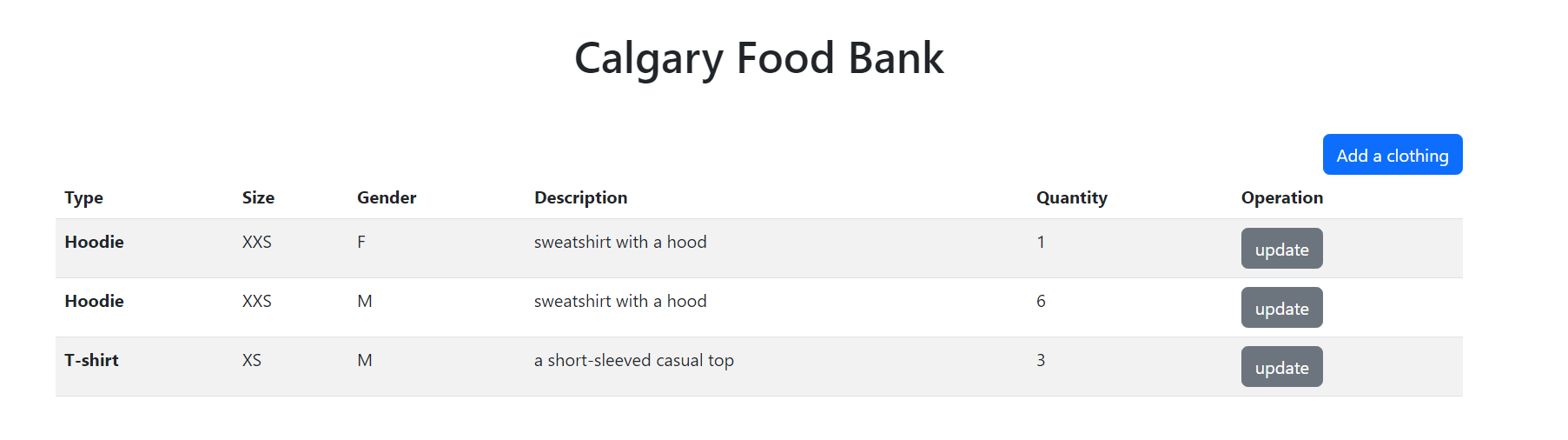
When the supervisor updates a selected food, the selected food will be bolded in a text field, and the supervisor inputs a new value to update the quantity of the food. Only numbers can be inputted, and trying to input anything else will not go through and the input field will remain empty. If the user presses update quantity while the input field remains empty, then nothing will happen and the page will remain. Once the user inputs a valid input and presses update it will go through and go back to the previous page and update the new quantity of the food. If the newly added food was not used in an order yet and you set the quantity to zero, it will be deleted from the database, otherwise the food remains but with a quantity of zero.

#### Adding new food



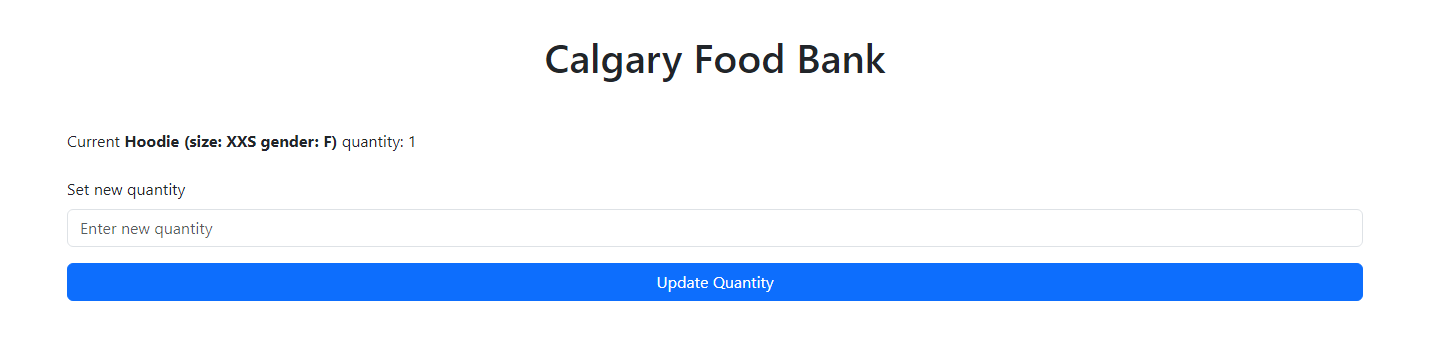
Pressing the “Add new food” button from the “replenish food inventory” page will take the user to a new page, where the user adds a new food to the database. The user inputs the food name, the calorie for the food, and the initial quantity of the food. Also, the user selects what type of food it is under. If any one of the input fields remain empty on submission, then the page will throw an error message and not add the food to the database and refresh the page. If the name of the food already exists in the database, an error will be thrown. Once the food has been added successfully, a message will pop up notifying the user that the food has been added, and it will be added to the database as well. Note that quantity cannot be zero and the supervisor responsible for this item will be the supervisor that created it.

#### Replenish clothing inventory page



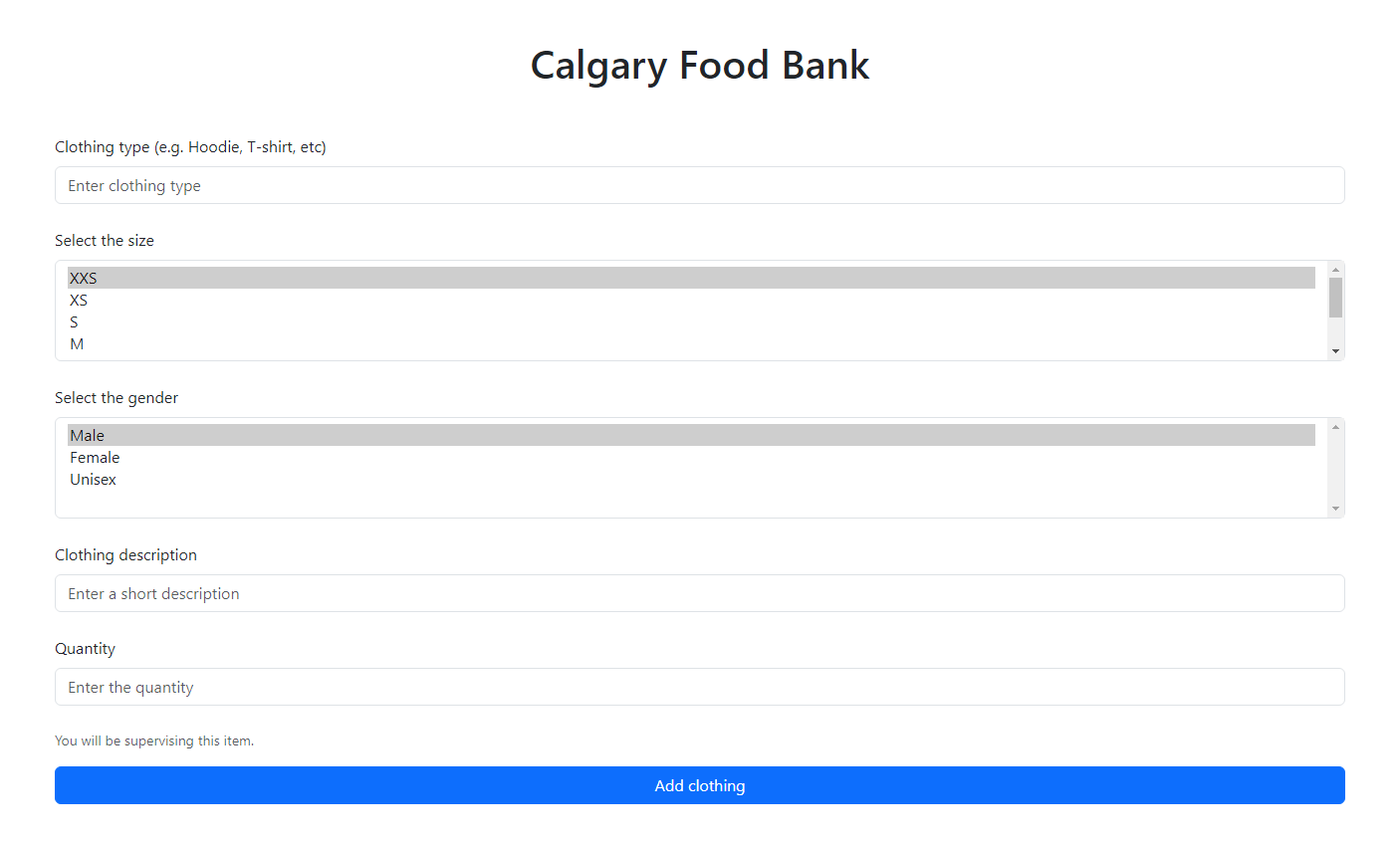
Replenish clothing inventory works the same as replenish food inventory except that it is adding, increasing, or decreasing in the clothing database. All of the functionalities of the buttons work the same as replenishing food inventory.

#### Update clothing quantity



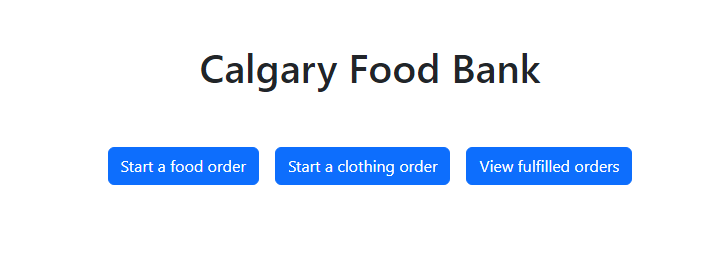
Pressing the “update” button from the “replenish clothing inventory” page will take the user to a new page, where the user can update the quantity of the item selected. The selected item will be bolded in the text field, and like updating the food quantity, the user can only input numbers into the input field. If the field remains empty, then the button will not do anything and does not change anything in the database. Once a valid number has been inputted and the user presses update, then the page will go back to the previous page, and show the change in the list as well as update the new quantity in the database. If the newly added clothing item was not used in an order yet and you set the quantity to zero, it will be deleted from the database, otherwise the item remains but with a quantity of zero.

#### Add new clothing



Pressing the “add new clothing” button from the “replenish clothing inventory” page will take the user to a new page, where the user can add a new clothing item to the database. In order to add a new clothing item, the user must fill in all of the input fields, where it is the clothing type, description, and the initial quantity. Not doing so will throw an error message to the user and not change anything in the database, and refresh the page. If the type, gender, and size of the clothing item already exists in the database, an error will be thrown. If adding new clothing does go through, a message will pop up notifying the user that the clothing has been added successfully, it will update the database and take the user back to the previous page. Note that quantity cannot be zero and the supervisor responsible for this item will be the supervisor that created it.

### Front Employee

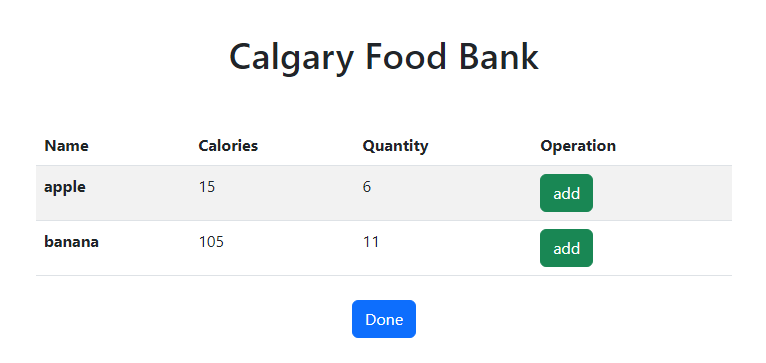


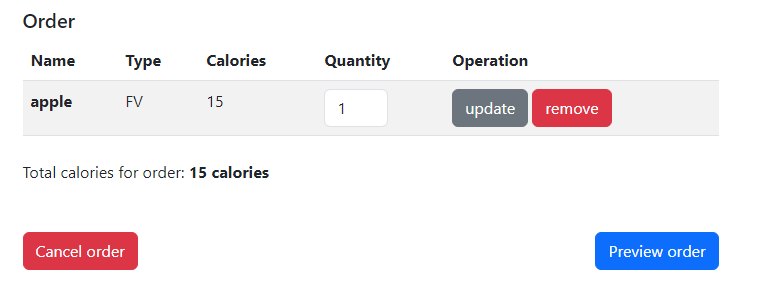
This is the home page for the front employee, and it consists of three buttons, Start a food order, start a clothing order, and view fulfilled orders:

#### Start a food order page



When the user presses start a food order, they are taken to this page, where the user can fill out a new order form. The form consists of filling information for the family, and each family member can range from 0 to 10 (children 0-10, adults 0-5). Once the user updates the family member, if the user presses the update family button, the bolded text field of the total calories will update as well. Then the user can select which type of food to add, where it consists of the types of food that exist currently in the database. For example, if the inventory only consists of fruits and vegetables, then the type of food that the user can select is only fruits and vegetables. The preview button will not work unless an item is added to the order. The “cancel order” button will direct the user back to home and cancel the order.



When the user selects a certain food type, it will take the user to a new page, where only the foods that are under the food type will show up in the list. If the user presses “add” on a certain food, it will be added to the food order and pressing done will take the user back to the order page with the updated food order. After clicking the add button, it will turn to a remove button so that the user can remove it from the order if needed.

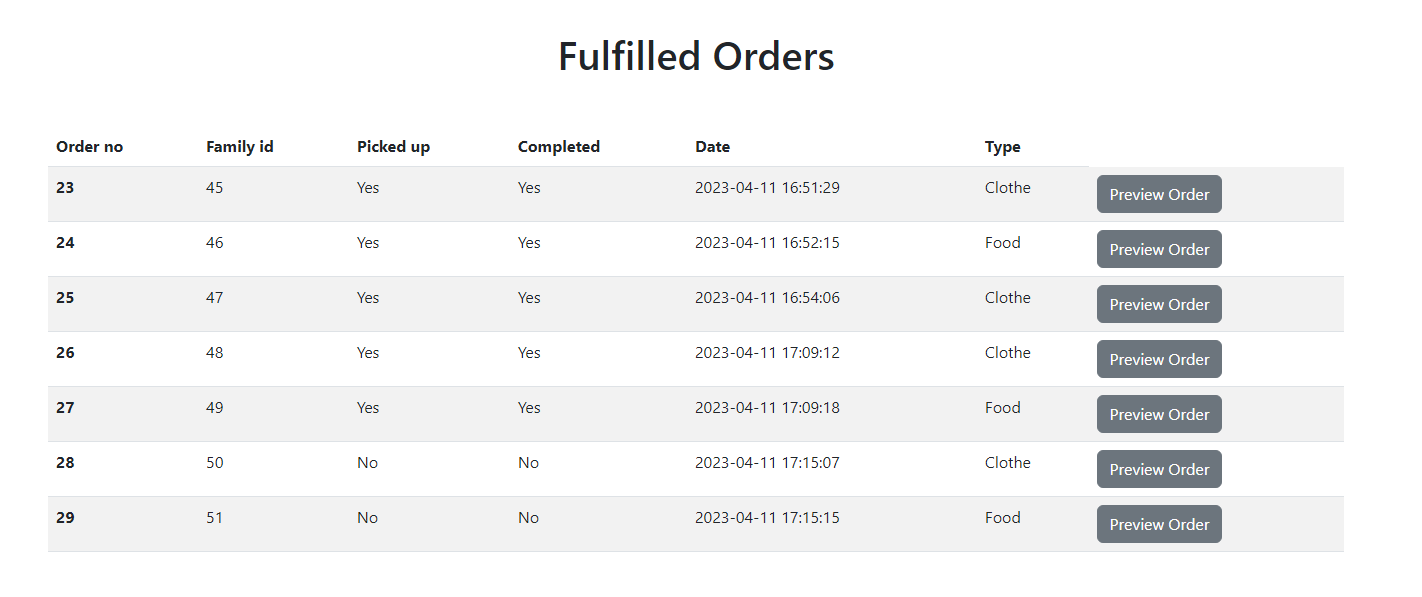
Once a food has been added, the user can select how much food they want to add, and once they select the quantity, they can press update, which updates the bolded calories text field below. The quantity of all added items is set to 1 by default.

#### Start a clothing order page

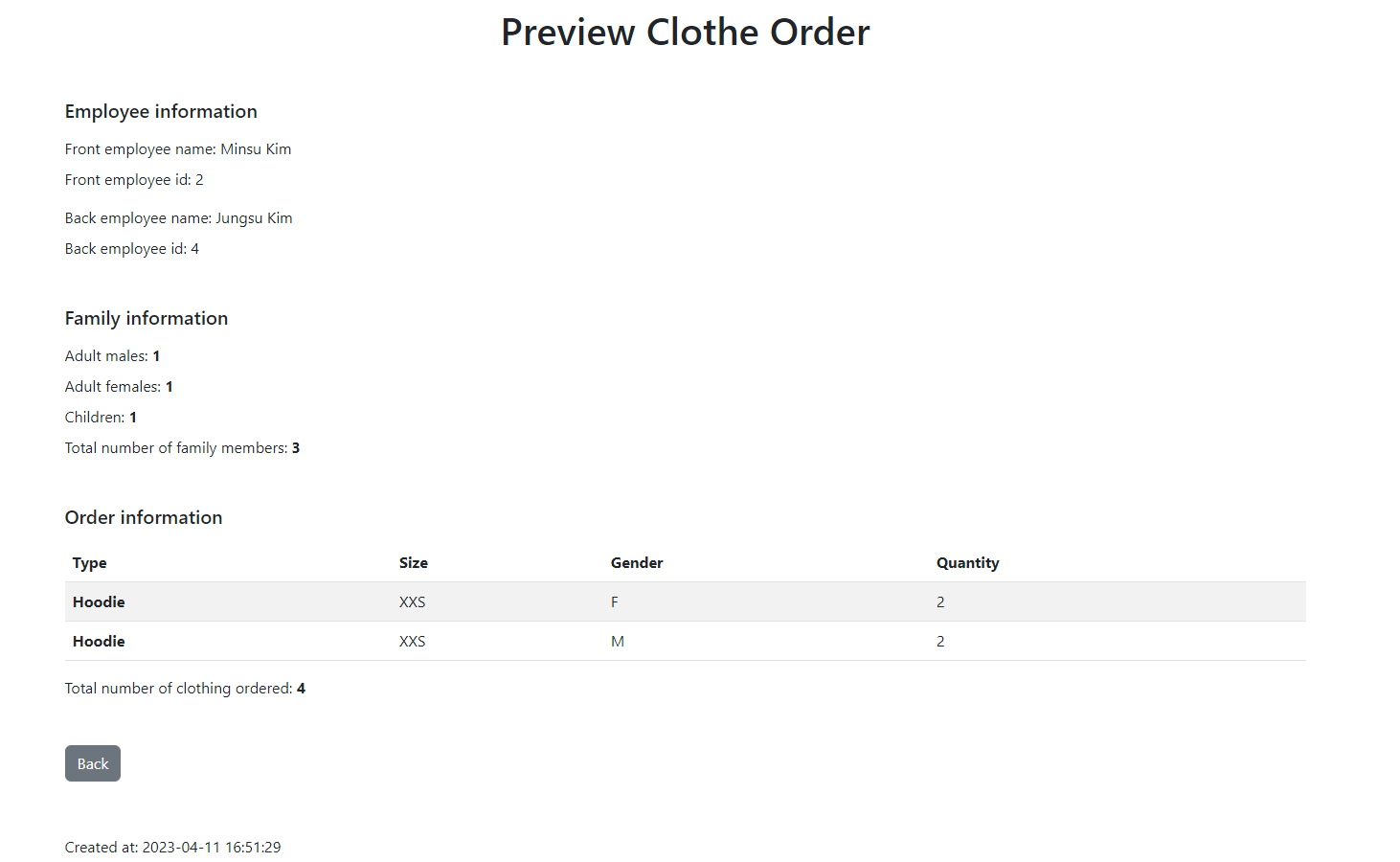


When the user presses start a new food order button, the user will be taken to a new page of clothing order, and the process works the same as a food order, where if the user updates the family members, it shows how many items the family should get, and the clothing types consists of the types of clothing that are available in the inventory, and the user can add, remove, and get multiple items.

#### View fulfilled orders page

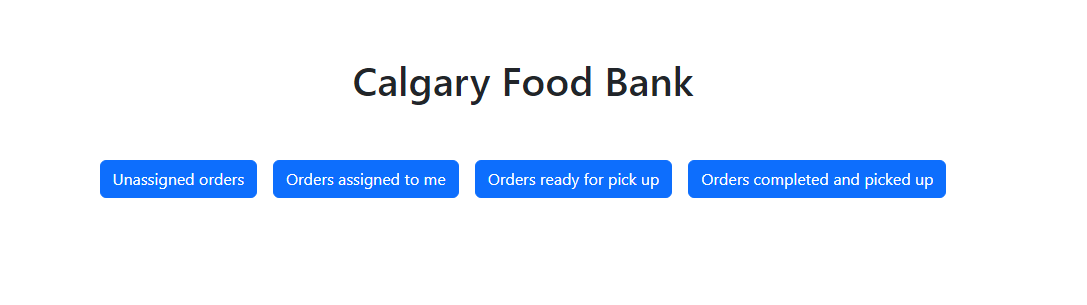


Fulfilled orders button takes the user to a page of all the orders that are placed. The user can press the preview order button which allows the user to see the order details.



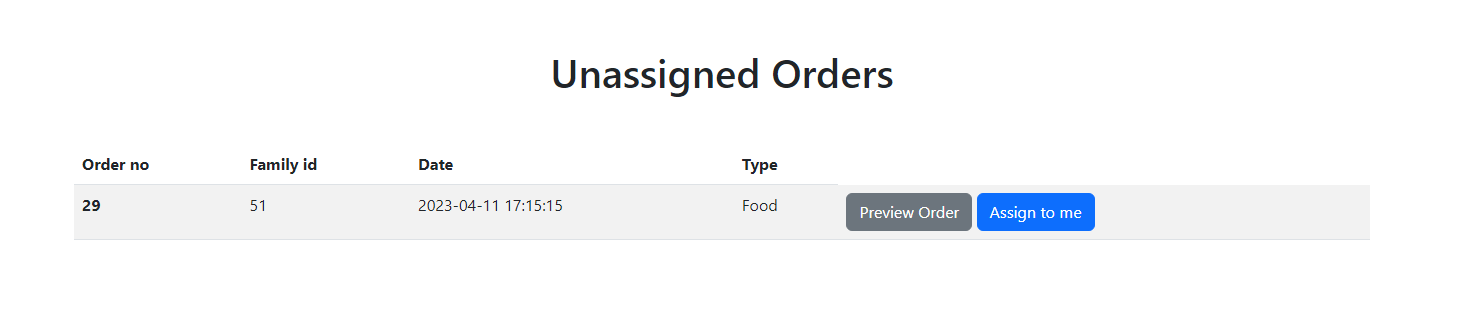
This is an example of an order preview, where the user can see which front employee has placed the order, and which back employee is fulfilling the order. Also, it shows the items of the order, and the date and time that the order has been placed at.

### Back Employee

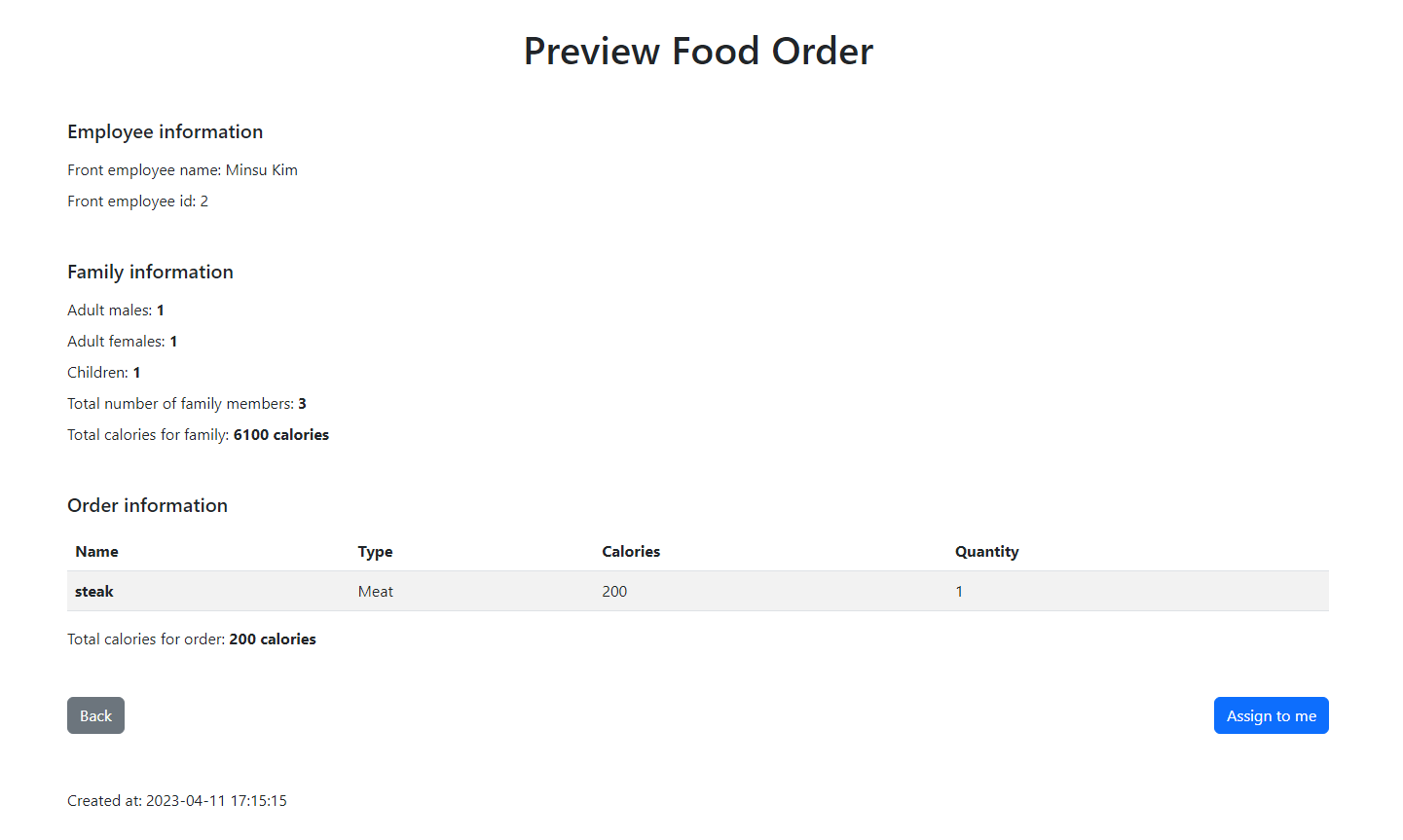


This is the home page of a back employee user. The home page consists of 4 buttons, unassigned orders, orders assigned to me, orders ready for pick up, and orders completed and picked up.

#### Unassigned orders page

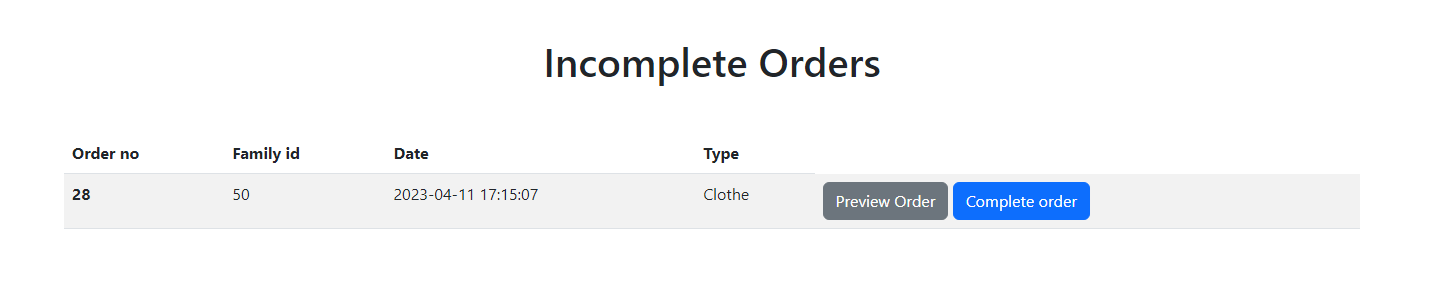


Unassigned orders page shows a list of orders that a back employee can choose to take, where no back employee is assigned to the orders. When a user enters this page, they can either preview the order, which leads the a page similar to the preview order page from above, however, the user also will be given a button that allows the user to assign the order to themself, or assign the order to themselves, where the order will be taken off the list once it has been clicked, and the order will be assigned to the user.

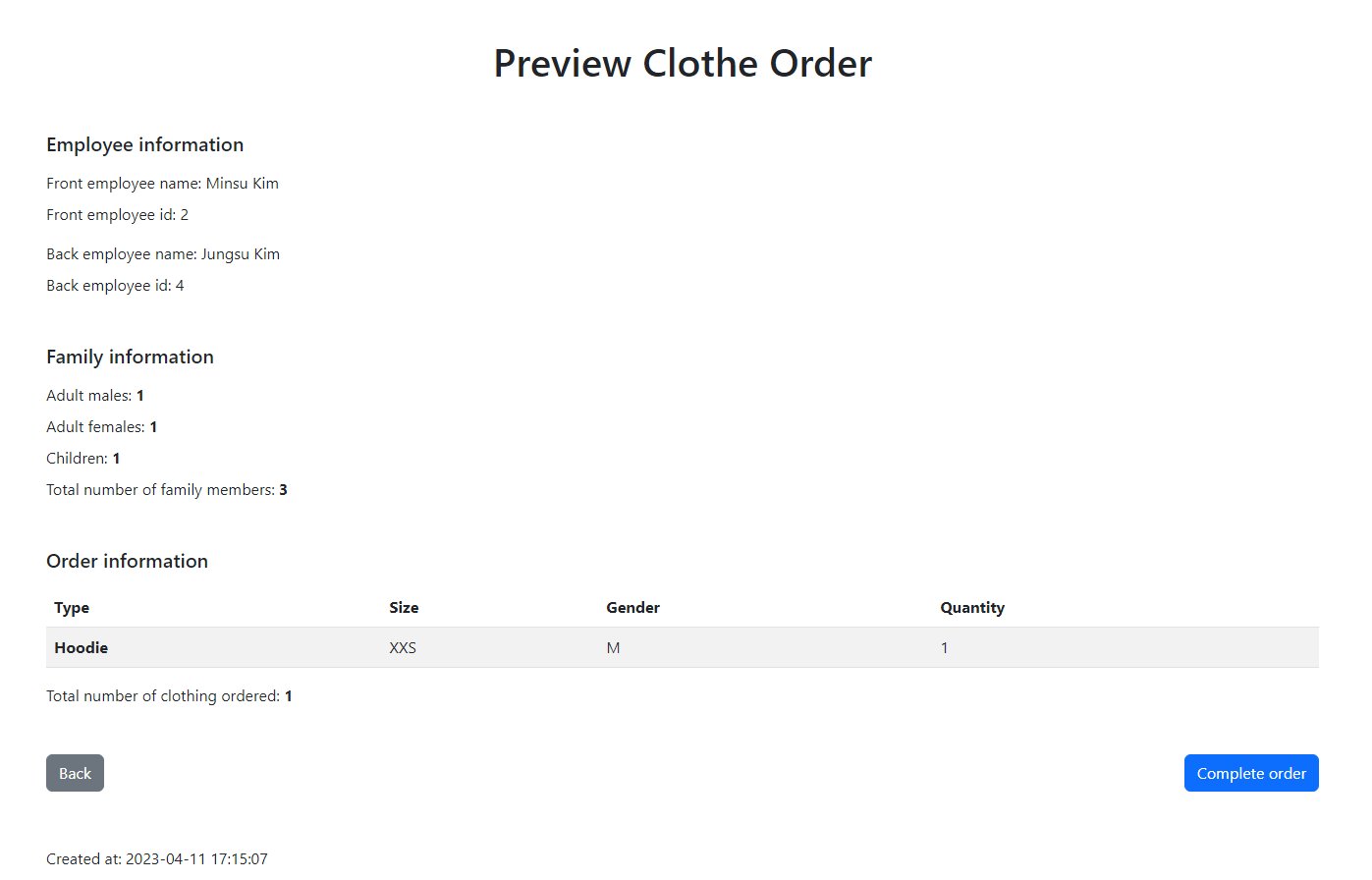


This page is from pressing the “preview order” button from the page above.

#### Orders assigned to me page

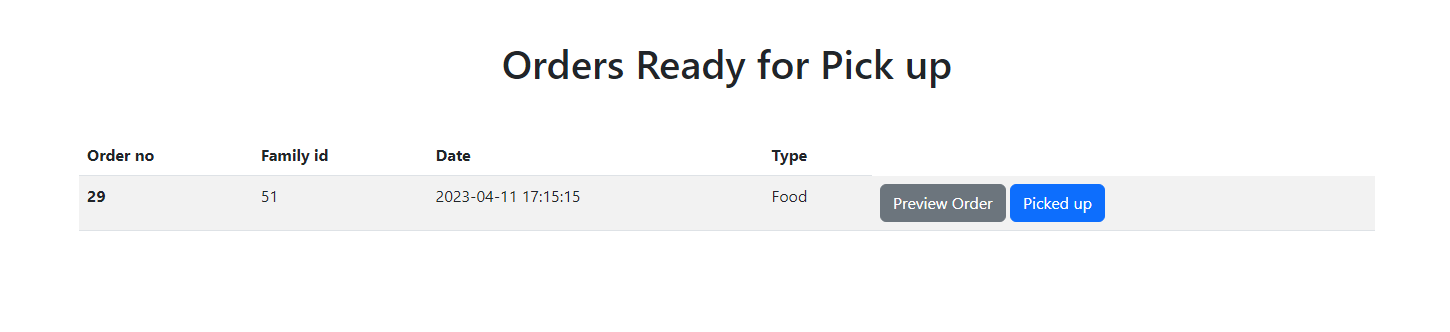


Orders assigned to me initially will show a list of orders that are only assigned to the specific user/employee. These are orders that are currently in the process of packaging. When the employee has finished completing the order he/she can then press the complete order button, the order is ready to be picked up, and is reflected in the database as well. The preview order sends the user to a page where they can see a further detailed version of the order, or be given an option to complete the order in that page as well, or the user can press complete order from the current page itself.

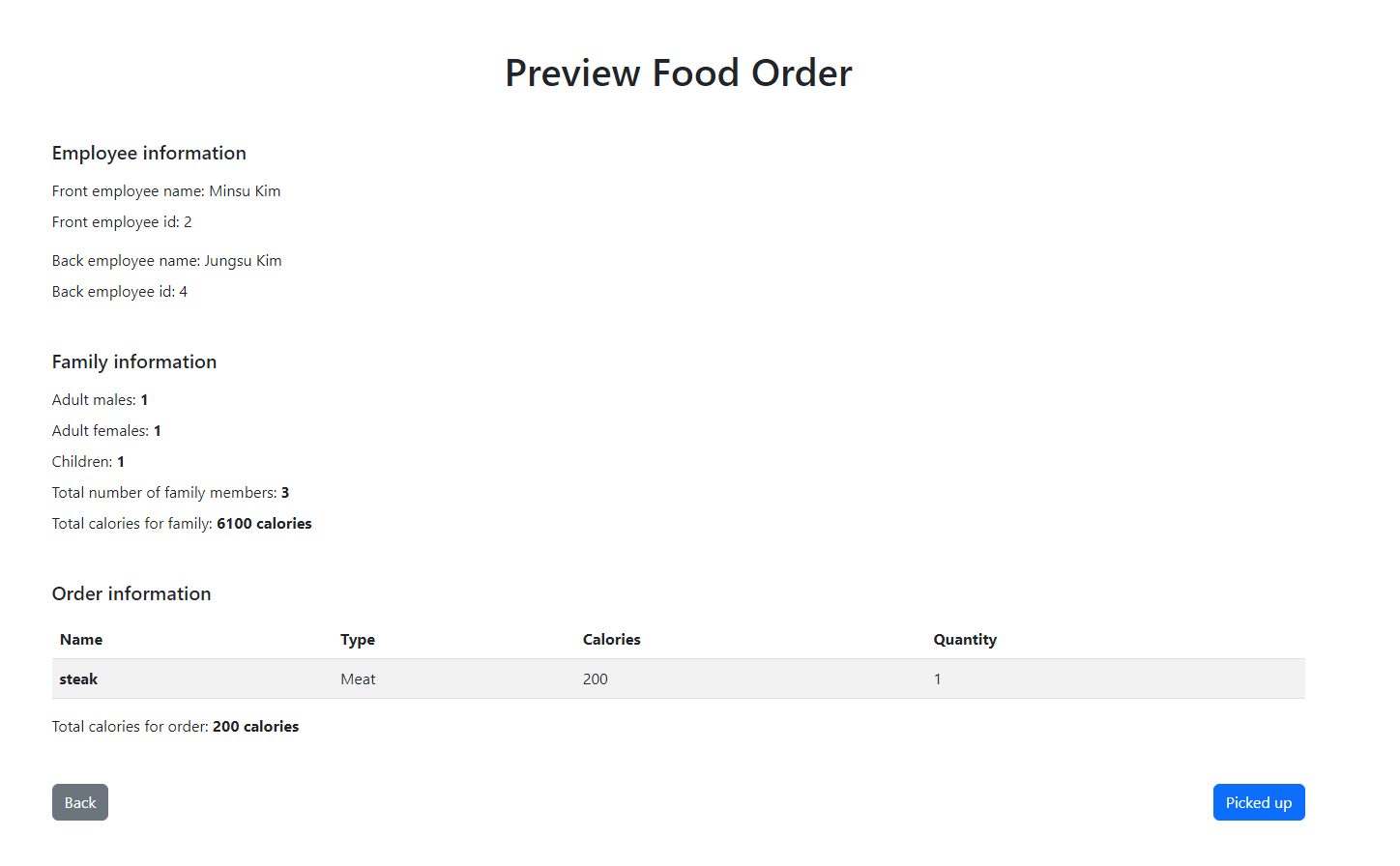


The picture above is an order that is shown when a user presses the “preview order” button from the “orders assigned to me” page.

#### Orders ready for pick up page

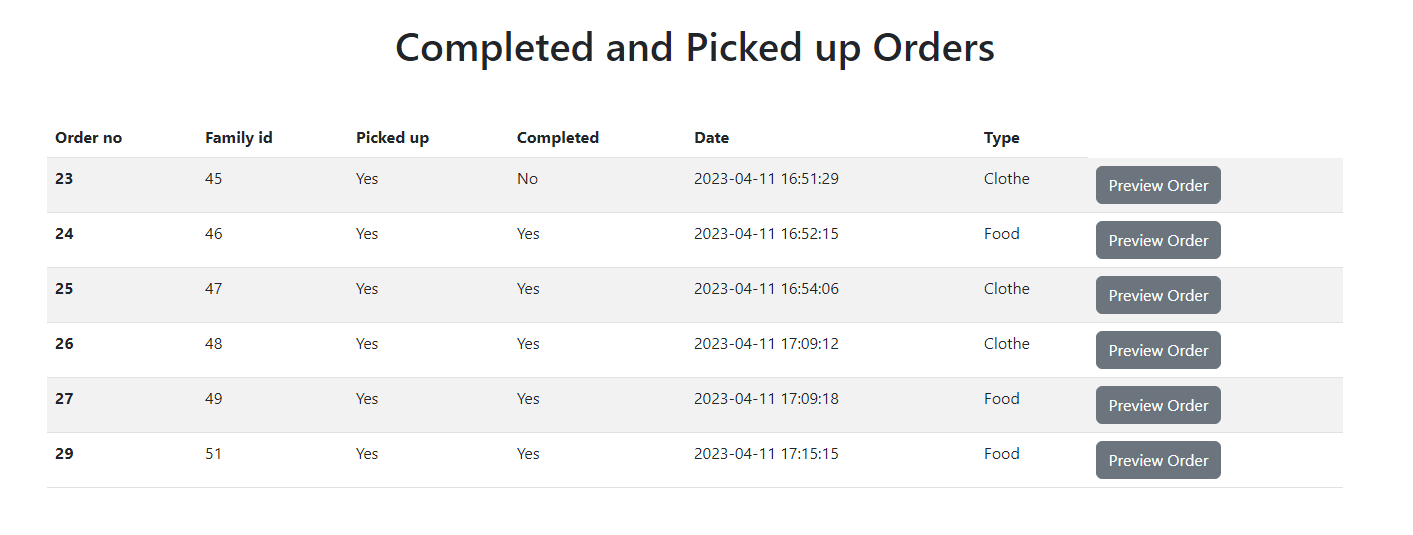


Orders ready for pick up page shows a list of orders that are marked as complete, and are ready to be picked up by the clients. The user can press the “preview order” button that leads the user to a different page that shows the details of the order, and a “picked up” button that the user can press once an order has been picked up by a family.

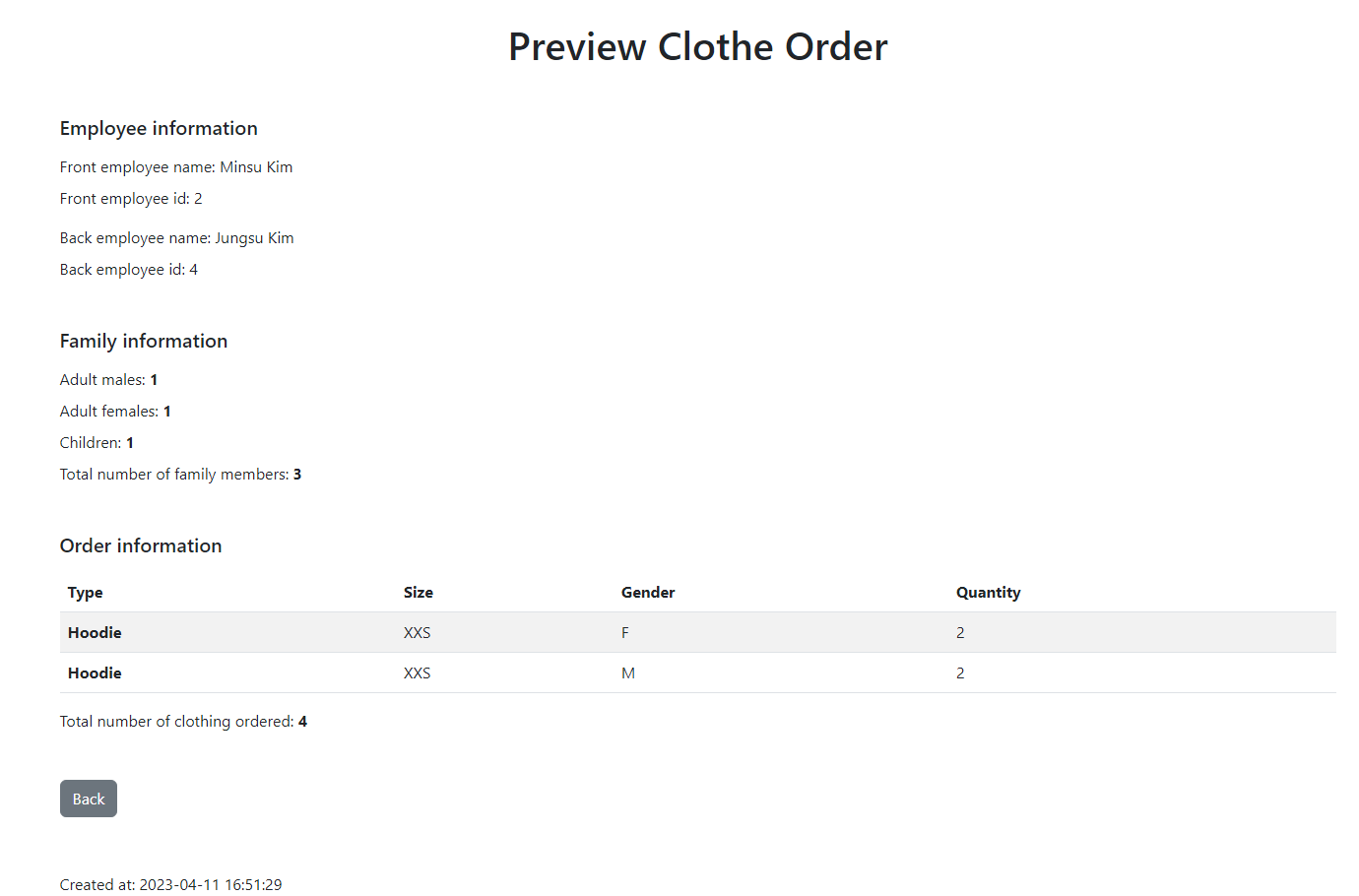


The picture above is the page that is directed to when a user presses the “preview order” button from the “orders ready for pick up” page, and in this page, the user is also given a “picked up” button as well.

#### Orders completed and picked up page



Orders completed and picked up page shows a list of orders that were completed and picked up. The user can press the “preview order” button that allows the user to see the order in detail.

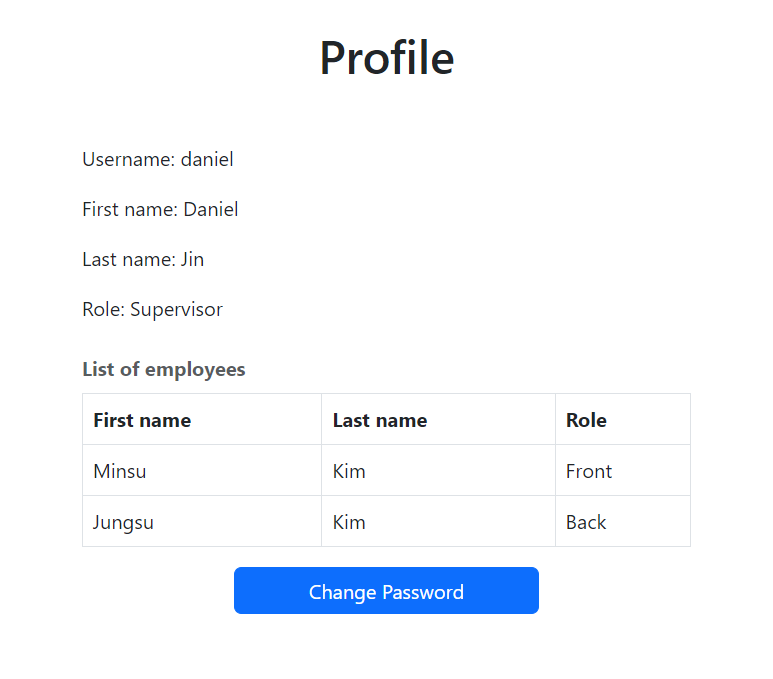


The picture above is the page that is directed to when a user presses the “preview order” button from the “orders completed and picked up” page.

## Profile page

At almost every page, the user will be able to access their profile by clicking on the drop down located at the top right corner of the page. The drop down has the first and last name of the current user at the top. It is then followed by a bolded text showing what role they have, and then a button that leads them to their profile page. Lastly, a red logout button which logs the user out and directs them back to the sign in page. Each profile shows different information depending on the role.

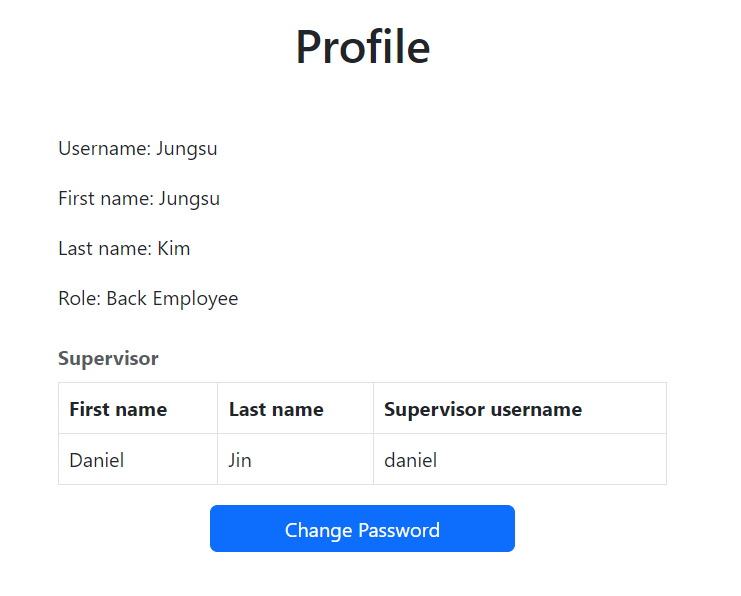
### Supervisor profile page



The “username” displays the username of the current user. The “First name” and “Last name”displays the first and last name of the current user, respectively. The “role” displays the role of the current user.

For a supervisor’s profile page, it will show the list of employees that the supervisor supervises. Finally, the change password button directs the user to a page that allows the user to edit their password.

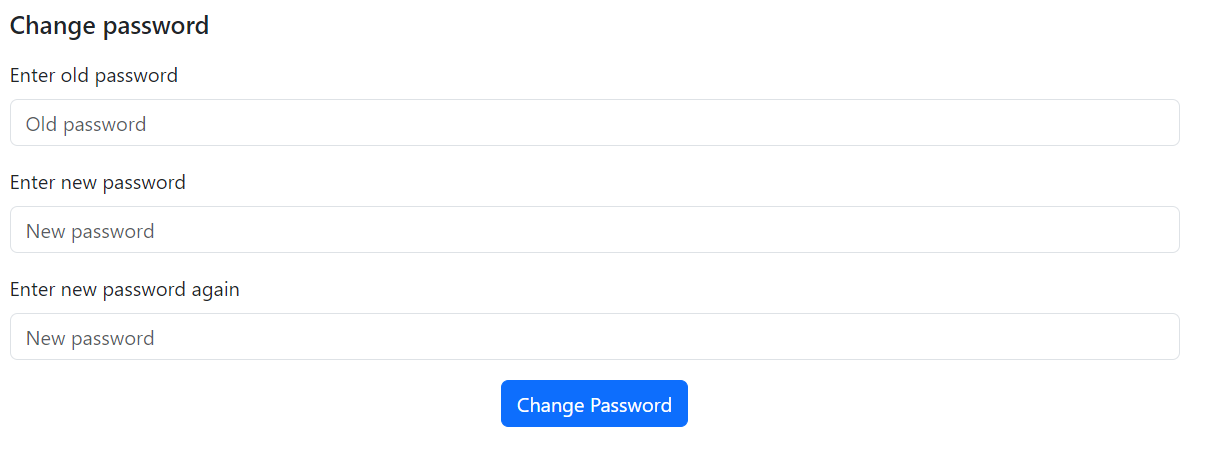
### Front/Back employee profile page



The “username” displays the username of the current user. The “First name” and “Last name”displays the first and last name of the current user, respectively. The “role” displays the role of the current user.

For a front and back employee’s profile page, it will show their supervisor. Finally, the change password button directs the user to a page that allows the user to edit their password.

### Change password page



In the change password page, in order for the user to change their password, the user must enter their current password, enter their new password twice and click the “change password” button. If any of the entries do not match the requirements or if the entries are empty, an error will be alerted. If all entries match the requirements, the password for the user will be changed.