Xin Yan Xie 10160975 T02

Rahmanta Satriana 30044533 T01

Alan Fung 10083599 T01

Food Bank Web APP Technical Report

CPSC 471 Final Report

**Abstract**

The SU Campus Food Bank, administered through SU Volunteer Services, provides

seven days a week emergency food relief service to the campus community.  Clients request food hampers by filling out forms with a volunteer in person, then selecting a time to pick up their hamper.  The key functionality of the food bank implemented in a database system, allowing for statistic tracking.

This report details work completed by the food bank team over the semester:

1. Review of the proposed database system key features
2. Overview of application design and user interaction
3. ERD diagram of food bank application
4. Translated relational schema, with justification for changes during implementation
5. Details on function implementation and the DBMS selected
6. Rationale for application functionality and SQL queries with pseudo-code
7. User manual
8. Appendix including database, backend and front code

**Introduction**

**Problem Definition**

In March of last year, the Student Union reported a shortfall (loss) of $30 000 on the food  
bank. In an interview with The Gauntlet, the food bank coordinators revealed that demand  
doubled in the previous two years, and is continuing to increase.  The food bank is

run by volunteers, who fill out paper forms detailing client information and item requests.  Because all information is kept on paper, any statistics besides cost are difficult to report, as this would require manual tabulation over hundreds of forms.

**Objectives**

This project aims to improve volunteer time usage and allow for additional insight into food bank usage by digitizing relevant information, primarily through inventory and request tracking.

Because all forms are manually filled in, it’s required that a volunteer be with clients when requests are made, and that they manually review forms for errors.  Digitizing these forms and implementing error checking would free up volunteers for other tasks, such as fundraising. As all forms are being stored electronically, the details in the requests can be used to track specific item usage over time, which would be time consuming in a paper based system.

**Project scope**

As detailed in the original proposal, the project deliverables are as follows:

* A web application that can be accessed online (statically hosted in the submission)
* A login system accommodating three types of users: admins, volunteers, and clients
* A system to assign volunteers to client requests
* Storage of client, admin, and volunteer information
* Storage of client request details
* Food inventory tracking
* Selected statistic reporting

Implementation details such as password hashing and user session cookies were considered out of the scope of this project, and as such are not present in the project implementation.

**User Interaction Overview**

Three types of users are defined in the database system:

1. Clients, who fill out request forms, request a pick up time, and receive food
2. Volunteers, who process requests, pack request items together, and are present at appointments where clients receive food items
3. Admins, who order food items from a supplier, and are responsible for teaching and management of volunteers

An example transaction with a client as it pertains to the completed application is as follows:

1. Client accesses web application, where they are asked to provide personal information and why they are requesting support for food
2. Client logs into the system, granting them access to the request page
3. Client fills in the request page, detailing which quantities they want of a predetermined list of foods
4. Volunteer logs into the system with credentials provided by their administrator
5. Volunteer accesses the ‘requests’ page, where a table of all uncompleted requests is presented
6. Volunteer selects a request to complete, accesses the ‘appointments’ page, and enters an appointment with the client, after which they pack the appropriate food items
7. Volunteer finishes packing food items
8. Volunteer attends appointment with client, and administers hampers
9. Volunteer accesses the ‘finish order’ page to update the request form and indicate that it has been completed which updates the food inventory in the database

Beyond this, admins can add and delete any level of users, appointments, orders, and update food inventories, funds as well as view the statistics of the database

1. To add new users, the admin can access the appropriate page in the ‘control panel’ page and enter in the relevant login credentials
2. To delete users, the admin would type in the relevant account ID of the user they wish to delete

**DBMS Selection Rationale**

This project was developed using the Flask Python Framework and Python3’s built-in Sqlite3 module. Flask was chosen for its built-in development server and debugger, as well as its minimalistic nature, allowing for great flexibility and control over what modules are used. The following imports were used for the backend application:

* Flask
  + Flask
  + render\_template
  + request
  + redirect
* sqlite3
* datetime

Sqlite3 was chosen for many reasons:

* It is a built-in module of Python3
* Sqlite code is extremely readable
* It has better performance, reduced application cost
* It is portable across all operating systems, 32-bit and 64-bit
* It is reliable, as content can be updated continuously and atomically so that little or no work is lost in a power failure or crash.
* The queries are similar to what was taught in class

**SQL Queries with associated pseudocode**

The function definitions in the backend application use Python to supplement SQL queries, typically completing tasks such as:

* Row validation
* Argument preprocessing
* List comprehensions on rows

Function name: reqform()

Function purpose: Returns the request form web page with a reference table of maximum requestable items

Pseudocode + SQL:

SELECT \* FROM MaxRequests

Embed MaxRequets in HTML page

Return webpage

Function name: loginAccount()

Function purpose: Logs the user into their appropriate accounts using authentication

Pseudocode + SQL:

If attempted login as client:

SELECT \* FROM Client

Parse SQL query results for client

If client account exists:

Set login as client

Return to homepage logged in as client

Else if attempted login as volunteer:

SELECT \* FROM Volunteer

Parse SQL query results for volunteer

If volunteer account exists:

Set login as volunteer

Return to homepage logged in as volunteer

Else:

SELECT \* FROM Admin

Parse SQL query results for admin

If admin account exists:

Set login as admin

Return to homepage logged in as admin

Function name: addreqform()

Function purpose: Submits a request form

Pseudocode + SQL:

SELECT \* FROM Dependant WHERE clientid = logged in user id

Calculate family size

SELECT \* FROM MaxRequests WHERE famSize= Calculated family size

Compare requested items to maximum requestable items

If requested appointment time is at least 1 day from today:

If requested items within maximum:

INSERT INTO RequestForm(fooditems,values)

\*\*note that the above insertion has over 14 arguments containing specific food items, and was redacted to maintain clarity

Function name: addAppointmentData()

Function purpose: Add an appointment between a client and a volunteer into data base

Pseudocode + SQL:

If appointment time is at least 1 day away:

If volunteer exists and client exists:

INSERT INTO Appointment (time, clientid, volunteerid ) VALUES ('{time}', '{clientid}','{volunteerid}').format(

time=request.form['time'],

clientid=request.form['cID'],

volunteerid=request.form['vID']))

\*\*the above values are submitted from the appointment webpage, formatting is as specified in Sqlite3 docs

Function name:  addInventoryData()

Function purpose: Add food store information

Pseudocode + SQL:

INSERT INTO Foodstore (refcode, foodname, quantity, expirydate, address) VALUES ('{refcode}','{foodname}','{quantity}', '{expirydate}', '{address}')".format(

refcode=request.form['refcode'],

foodname=request.form['foodname'],

quantity=request.form['quantity'],

expirydate=request.form['expirydate'],

address=request.form['address']))

Function name: register()

Function purpose: Add a client account to the database

Pseudocode + SQL:

INSERT INTO Account (name, email, username, password, accounttype) VALUES ('{name}','{email}','{username}', '{password}', '{accounttype}').format(

name=request.form['name'],

email=request.form['email'],

username=request.form['username'],

password=request.form['password'],

accounttype=0))

INSERT INTO Client (income,accountid) VALUES ('{income}','{accountid}')".format(

income=request.form['income'],

accountid=account\_id))

For all dependants:

INSERT INTO Dependant (clientid,name,relationship) VALUES ('{clientid}','{name}','{relationship}')".format(

clientid=account\_id,

name=request.form[varname],

relationship=request.form[relname]))

For all reasons:

INSERT INTO Reasons (clientid,reason) VALUES ('{clientid}','{reason}')".format(

clientid=account\_id,

reason=request.form[reasonname]))

Function name: addAdminUser()

Function purpose: Add an admin account to the database

Pseudocode + SQL:

INSERT INTO Account (name, email, username, password, accounttype) VALUES ('{name}','{email}','{username}', '{password}', '{accounttype}')".format(

name=request.form['name'],

email=request.form['email'],

username=request.form['username'],

password=request.form['password'],

accounttype=2))

INSERT INTO Admin (phonenumber,accountid) VALUES ('{phonenumber}','{accountid}')".format(

phonenumber=request.form['phone'],

accountid=account\_id))

Function name: addVolunteerUser()

Function purpose: Add a volunteer account to the database

Pseudocode + SQL:

INSERT INTO Account (name, email, username, password, accounttype) VALUES ('{name}','{email}','{username}', '{password}', '{accounttype}')".format(

name=request.form['name'],

email=request.form['email'],

username=request.form['username'],

password=request.form['password'],

accounttype=1))

INSERT INTO Volunteer (phonenumber,availability,accountid,managerid) VALUES ('{phonenumber}','{availability}','{accountid}','{managerid}')".format(

phonenumber=request.form['phone'],

availability=request.form['availability'],

accountid=account\_id,

managerid=request.form['mID']))

Function name: addDonorData()

Function purpose: Add a donor to the database

Pseudocode + SQL:

INSERT INTO Donor (name,phonenumber,email) VALUES ('{name}','{phonenumber}','{email}')".format(

name=request.form['name'],

phonenumber=request.form['phonenumber'],

email=request.form['email']))

Function name: addSupplierData()

Function purpose: Add a supplier to the database

Pseudocode + SQL:

INSERT INTO Supplier (name,phonenumber,email) VALUES ('{name}','{phonenumber}','{email}')".format(

name=request.form['name'],

phonenumber=request.form['phonenumber'],

email=request.form['email']))

Function name: updateFundsData()

Function purpose: Updates the funds of the Food bank

Pseudocode + SQL:

Get foodbank address

UPDATE Foodbank SET funds = request.form['funds'] WHERE address = foodbank address

Function name: viewAdmin()

Function purpose: Return a webpage with the list of admin users

Pseudocode + SQL:

SELECT \* FROM Account INNER JOIN Admin ON Account.id=Admin .accountid

Return webpage

Function name: viewVolunteer()

Function purpose: Return a webpage with the list of volunteer users

Pseudocode + SQL:

SELECT \* FROM Account INNER JOIN Volunteer ON Account.id=Volunteer.accountid

Return webpage

Function name: viewClient()

Function purpose: Return a webpage with the list of client users

Pseudocode + SQL:

SELECT \* FROM Account INNER JOIN Client ON Account.id=Client.accountid

Return webpage

Function name: viewAppointment()

Function purpose: Return a webpage with the list of appointments

Pseudocode + SQL:

SELECT \* FROM Appointment

Return webpage

Function name: viewInventory()

Function purpose: Return a webpage with the list of foodstores

Pseudocode + SQL:

SELECT \* FROM Foodstore

Return webpage

Function name: viewOrder()

Function purpose: Return a webpage with the list of request forms

Pseudocode + SQL:

SELECT \* FROM RequestForm

Return webpage

Function name: deleteAdminUser()

Function purpose: Delete an admin user from the database

Pseudocode + SQL:

Get account ID of the admin user

DELETE FROM Account WHERE id= accountid of admin user

Function name: deleteVolunteerUser()

Function purpose: Delete a volunteer user from the database

Pseudocode + SQL:

Get account ID of the volunteer user

DELETE FROM Account WHERE id= accountid of volunteer user

Function name: deleteClientUser()

Function purpose: Delete a client user from the database

Pseudocode + SQL:

Get account ID of the client user

DELETE FROM Account WHERE id= accountid of client user

Function name: deleteOrderData()

Function purpose: Delete a request form from the database

Pseudocode + SQL:

Get account ID of the request form

DELETE FROM RequestForm WHERE id= id of request form

Function name: deleteAppointmentData()

Function purpose: Delete an appointment from the database

Pseudocode + SQL:

Get account ID of the appointment form

DELETE FROM Appointment WHERE id= id of appointment form

Function name: genStatsVolunteer()

Function purpose: Generate a webpage with the statistics of a volunteer

Pseudocode + SQL:

SELECT COUNT(id) FROM Appointment WHERE ? = volunteerid, (request.form['Vid']))

Retuern Webpage

Function name: genStatsClient()

Function purpose: Generate a webpage with the statistics of a client

Pseudocode + SQL:

SELECT COUNT(id) FROM Appointment WHERE ? IN (SELECT accountid FROM Client, (request.form['Vid']))

Retuern Webpage

Function name: genStatsAllClient()

Function purpose: Generate a webpage with the statistics of all clients that are low income

Pseudocode + SQL:

SELECT COUNT(id) FROM Appointment WHERE clientid IN (SELECT accountid FROM Client WHERE income <= 10000)

Retuern Webpage

Function name: finishOrder()

Function purpose: Update request form with id of volunteer that completed the request, populate “Takes” table with clientid and quantities of items client received, update “Foodstore” with taken items.

Pseudocode + SQL:

Implementation incomplete

**User Manual**

In Linux:

cd CPSC471FoodBank

pip install flask

chmod+x run.sh

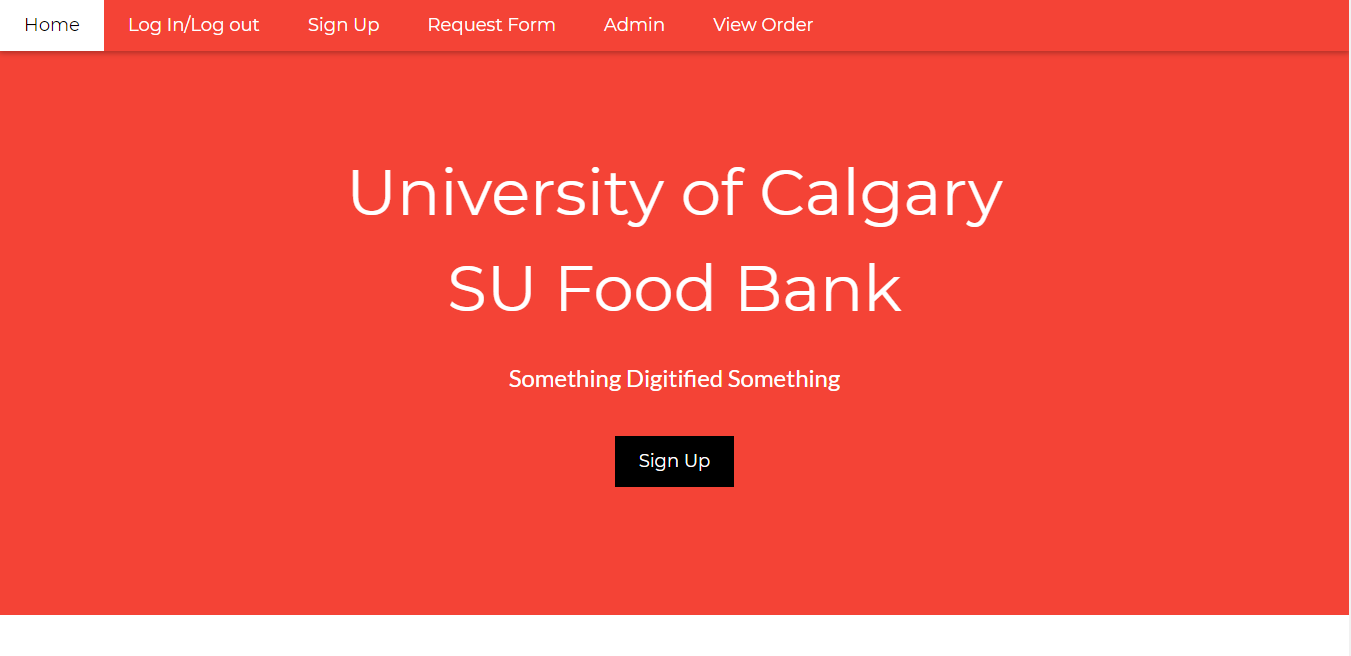
./run.sh

After executing the shell script, the web application will be statically hosted on the local machine. To access the web application, go to your preferred browser(except for Microsoft Explorer or Edge)

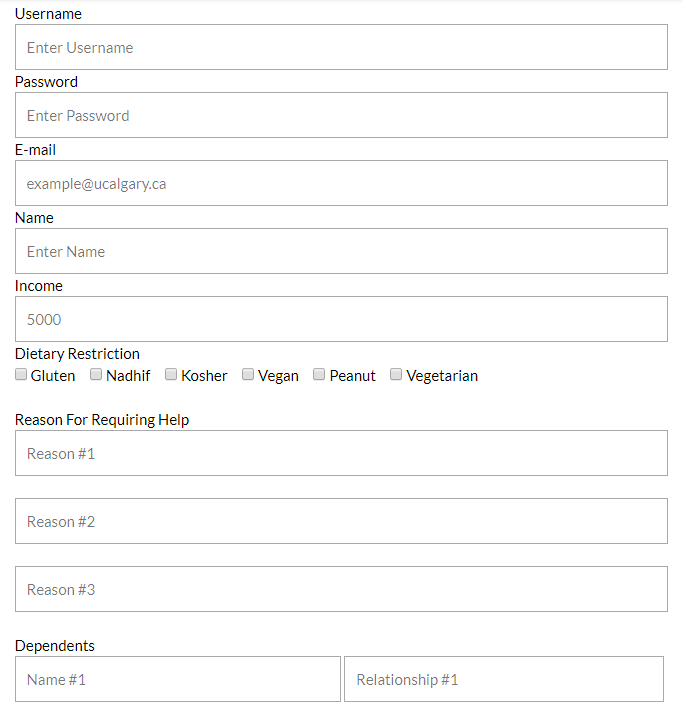
And enter the following address: localhost:5000



The first page is the home page and looks like so:



The first thing you must do is sign up:



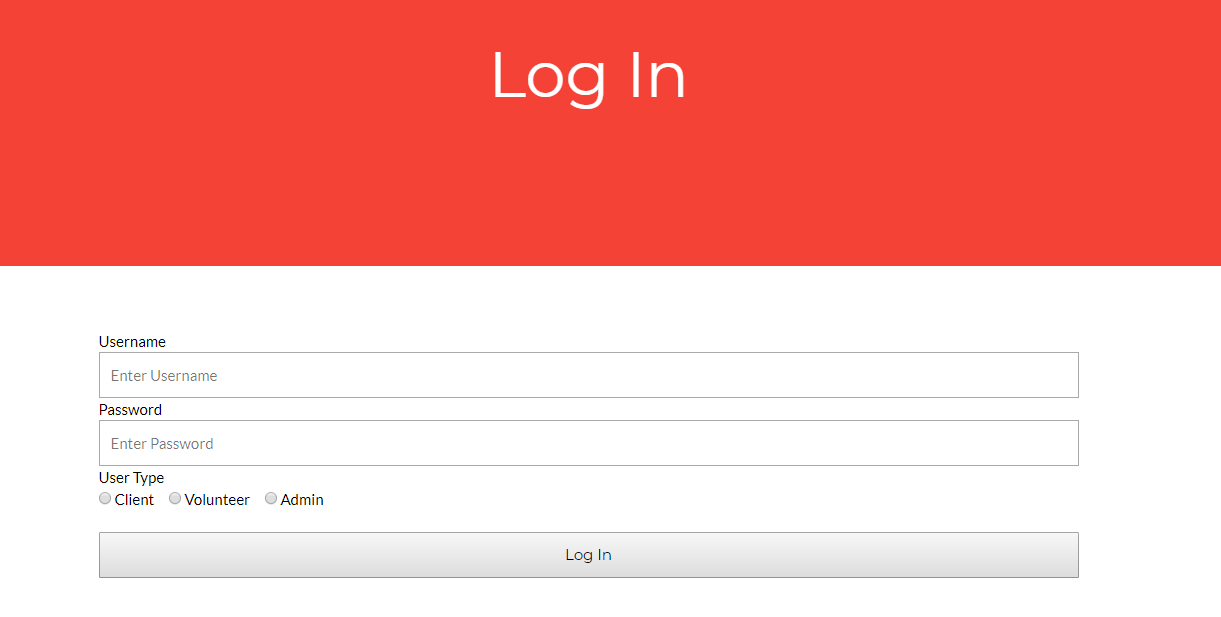
The format of the fields are enforced by the webpage

After signing up, you will be redirected to the home page,

You may now log in as a client using your credentials by pressing the Log In/Log out button on the navigation bar



Now enter your previously registered credentials



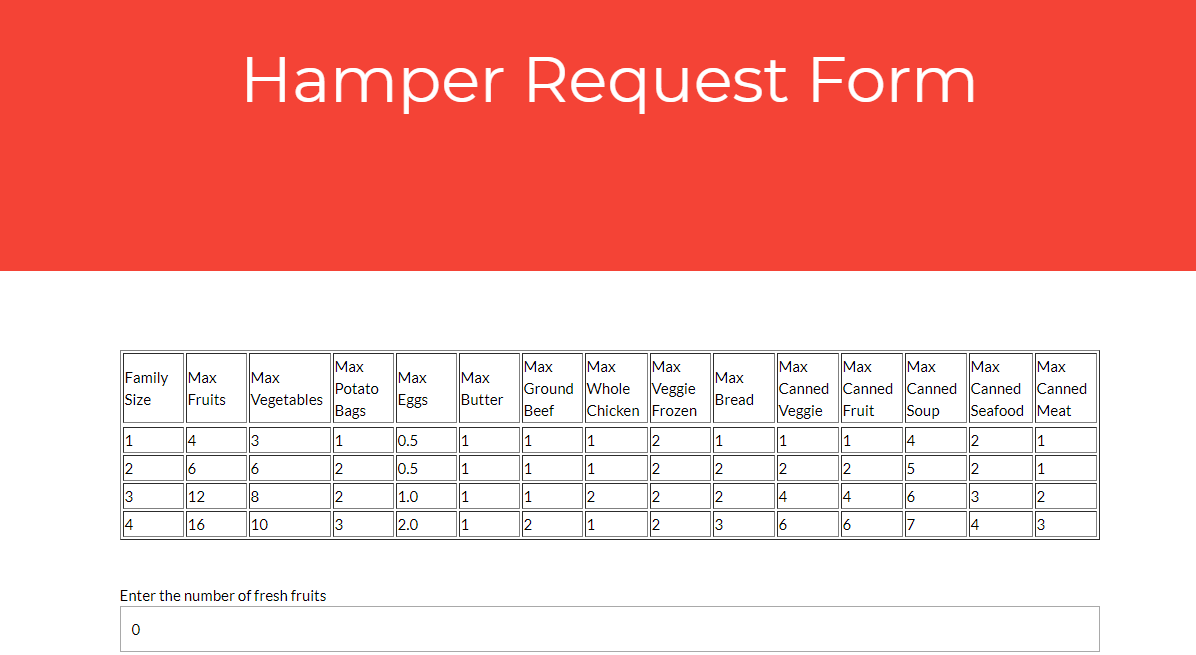
Note that you must select the Client radio button since you are logging in as a client

Note that if you are testing as a Volunteer or an Admin, the process is the same but you would select the appropriate radio buttons.

After logging in as a client, you will be returned to the homepage, and the request form page will be available to you.

Navigate to that page by pressing the Request Form button in the navigation bar



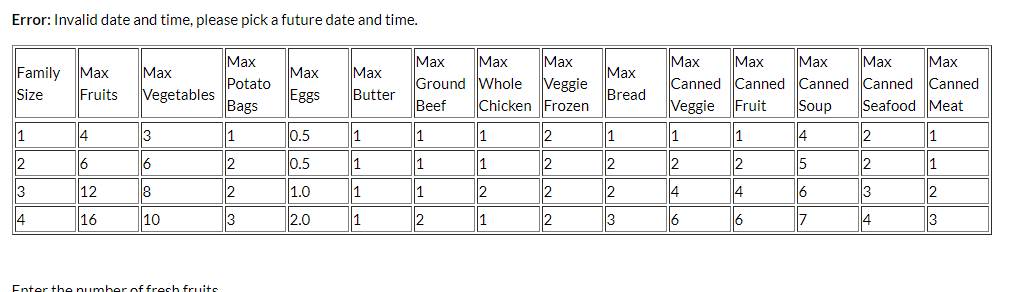


The table you see in the Hamper Request Form indicates the maximum of requestable food items of each type for the given family size.

Attempting to request more than the indicated maximum will result in an error and return you back to the Request Form page with all fields reset.

Additionally, the date of the request form must be valid, meaning it must be sometime in the future.

Below is an example of what an error looks like



If there were no errors, the request was successful

At this point, you are finished as a client, so you may press the Log In/ Log out button to log out to complete your session

Now lets proceed as a volunteer

Typically a volunteer would be provided credentials by an Admin

A test account has been made available

Navigate to the Log In page and use the following credentials to log in as a volunteer

Username: east

Password: 111111

Press the login button

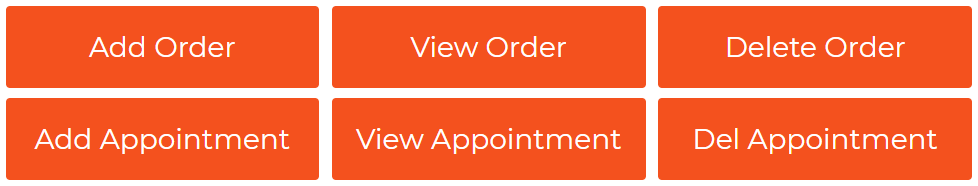
After successfully logging in as a volunteer, you will be returned to the home page

You will now have access to the control panel



As a volunteer, you are able to access some of these functions

Namely, any of the buttons that deal with Orders or Appointments



Attempting to access any other buttons in the control panel will do nothing and return you to the control panel page

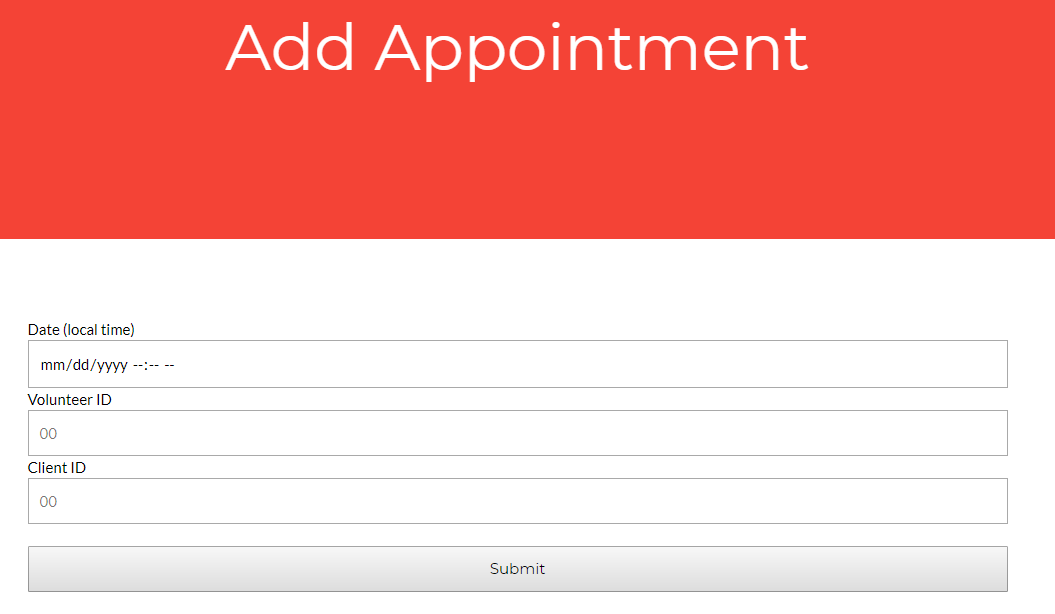
As a volunteer, one of your responsibilities is setting up appointments with clients as you process their orders.

Check which orders are pending by pressing the View Order Button



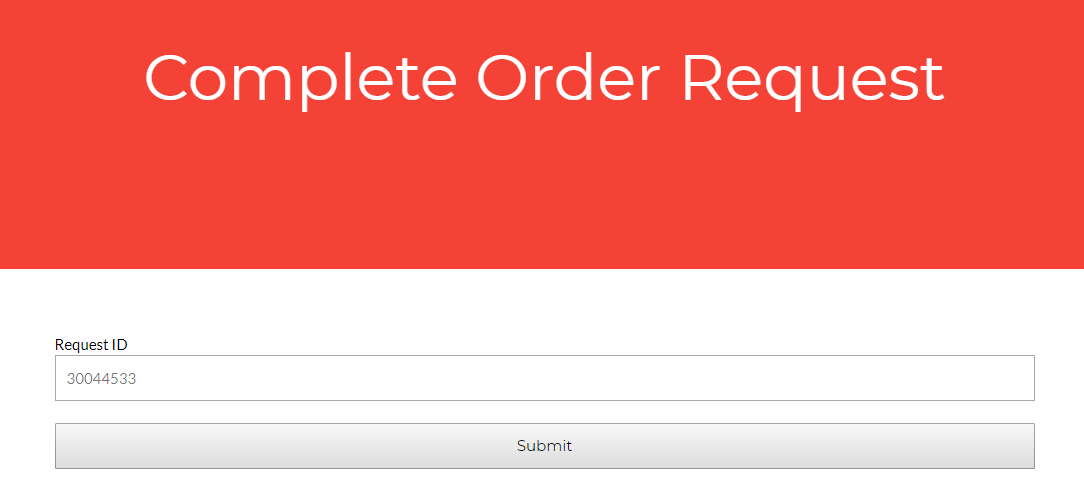
Note the Volunteer Handler ID column, if the value is None, the order is yet to be processed and you are encouraged to work on it.

Once you have chosen a request to work on, you must add an appointment with the respective client by pressing the Add Appointment button in the Control Panel page



Note the pickup date from the order request, as this is the client’s preferred time. You have the freedom to enter a different time for the pick-up appointment but are discouraged from doing so.

After successfully meeting with the client and once they have left with the hamper, you may mark the request form as finished by pressing the Finish Order button in the Control Panel and entering in the request form ID you have just completed.



Doing so will accomplish three things:

* Marking your volunteer ID in the Volunteer Handler ID column of the View Orders page so that others can see that request has been completed.
* Storing the quantities of the taken food items in the Takes Table
* Decrementing the respective quantities in the food stores to represent the taken food

That has covered the functionalities available to volunteers

At this point, you may log out and relog in as an Admin

A test account has been made available

Navigate to the Log In page and use the following credentials to log in as an admin

Username: csus

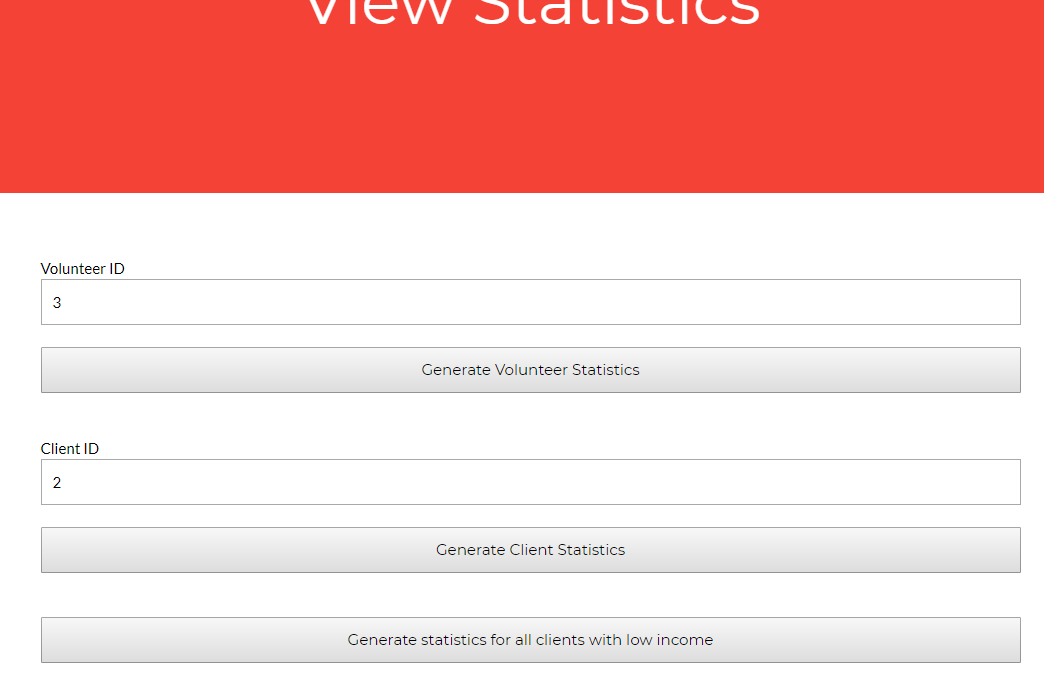
Password: cpsc2018

After logging in as an Admin, all buttons in the control panel will be made available

An Admin has the power to add and delete users, amongst other things.

Add and Delete are self explanatory.

To generate user statistics, press the View Statistics button



On this page, an Admin can view the number of appointments a volunteer or client has

**Appendices follow**