Phase 2 Abstract Code w/SQL
CS 6400 - Spring 2017
Team 090

Table of Contents

Landing Page

Show ***Login*** and ***Reports*** as tabs.
Upon:

- Click ***Login*** tab button - Opens a form for the **Authenticate User Info** task
- Click ***Reports*** tab button - The following dropdown list of menu options is displayed: ***Rooms/Bunks Availability***, ***Meals Remaining***
  - (1) Click on ***Rooms/Bunks Availability*** button - Opens a form for **View Room/Bunk Availability report** task
  - (2) Click on ***Meals Remaining*** button - Opens a form for **View Meals Remaining report** task

Authenticate User Info

Abstract Code

- User enters *Username*($Username), *Password* ($Password) input fields
- When ***Login*** button is clicked, if data validation is successful for both *Username* and *Password* fields, then attempt to locate a user in the system:

```
SELECT * FROM User WHERE Username= '$Username'' AND Password =
'$Password';
```

- If User record is not found or User.password != '$Password':
    - Go back to **Authenticate User Info** form, with error message
- Else:
    - Store login information as session variable $Username and $SiteID
    - Go to the **Main Menu** form

Main Menu

Abstract Code

Show ***Site details***, ***Items***, ***Clients***, ***Requests***, **and** ***Logout*** as tabs.

Upon :

- 1. Click of ***Site details*** tab: Open a form for the **Add/View/Update/Delete Service** task

- 2. Click of ***Items*** tab: Display the following list of menu options: ***Add Item, Search/Update/Request Item(s)***
  (2.1) Click of ***Add Item*** button - Open a form for the **Add Item** task
  (2.2) Click of ***Search/Update/Request Item(s)*** button- Open a form for the **Search/Update/Request Item(s)** task

- 3. Click of ***Clients*** tab: Display the following list of menu options: ***Add Client***, ***Client Search***, ***Wait List Report***
  (3.1) Click ***Add Client*** button - Open a form for the **Add Client** task
  (3.2) Click ***Client Search*** button - Open a form for the **Client Search** task
  (3.3) Click ***Wait List Report*** - Open a form form for the ***Wait List Report*** task

- 4. Click of ***Requests*** tab: Display the following dropdown list of menu options: ***Outstanding Requests***, ***Item Requests***

(4.1) Click on ***Outstanding Request*** button - Open a form for **Approve/ Edit / Delete Requests** task.
(4.2) Click on ***Item Requests*** button - Open a form for **View Item Request status** task.

- 5. Click of ***Logout*** tab button: Invalidate the login session and take the user back to the **Authenticate User Info** form.

Add, View, Update or Delete Service

Abstract Code

This task is used for Adding / modifying and deleting site service details.

- Check for the existence of each service type for the logged in user's $SiteID.

```
SELECT COUNT(SiteID) FROM FoodBank WHERE SiteID = $SiteID
SELECT COUNT(SiteID) FROM Shelter WHERE SiteID = $SiteID
SELECT COUNT(SiteID) FROM SoupKitchen WHERE SiteID = $SiteID
SELECT COUNT(SiteID) FROM FoodPantry WHERE SiteID = $SiteID
```

- For service types that already exist, show two buttons: ***View/Update*** and ***Delete***. (Except for the FoodBank service since there is nothing to update there.)
- For service types that do not already exist, show a button: ***Add***

Add service

- When ***Add*** button is clicked, show the **Add Form** for the related service type. The form will contain all the necessary fields for the given service type.
- Once the form is filled out correctly, user will click **Submit.** If valid input, insert a new record to the corresponding table using the logged in user's $SiteID. Otherwise show a validation message.

```
INSERT INTO soupkitchen (SiteID, TotalSeatsAvailable,
RemainingSeatsAvailable, HoursOfOperaion, ConditionsForUse)
VALUES ('$SiteID' ,$TotalSeatsAvailable, $RemainingSeatsAvailable
'$HoursOfOperaion', '$ConditionsForUse');

INSERT INTO shelter (SiteID, MaleBunksAvailable, FemaleBunksAvailable,
MixedBunksAvailable, RoomsAvailable, HoursOfOperaion, ConditionsForUse)
VALUES( '$SiteID', '$MaleBunksAvailable', '$FemaleBunksAvailable',
$MixedBunksAvailable ,$RoomsAvailable, '$HoursOfOperaion',
$ConditionsForUse');
```

```
INSERT INTO foodpantry (SiteID, HoursOfOperaion, ConditionsForUse)
VALUES ('$SiteID', '$HoursOfOperaion', '$ConditionsForUse');

INSERT INTO foodbank (SiteID) VALUES ('$SiteID');
```

Update service

- When the **Update** button is clicked, show the **Update Form** for the related service type. The form will contain all the necessary fields for the given service type.
- The form will be initially populated with all the existing values for the given service.

```
SELECT * FROM FoodBank WHERE SiteID = $SiteID
SELECT * FROM Shelter WHERE SiteID = $SiteID
SELECT * FROM SoupKitchen WHERE SiteID = $SiteID
SELECT * FROM FoodPantry WHERE SiteID = $SiteID
```

- Once the form is filled out correctly, user will click **Submit.** If valid input, update the existing record in the corresponding table using the logged in user's $SiteID. Otherwise show a validation message.

```
UPDATE soupkitchen set TotalSeatsAvailable='$TotalSeatsAvailable',
RemainingSeatsAvailable='$RemainingSeatsAvailable',
HoursOfOperaion='$HoursOfOperation', ConditionsForUse='$ConditionsForUse'
WHERE SiteID='$SiteID';

UPDATE shelter set MaleBunksAvailable='$MaleBunksAvailable',
 FemaleBunksAvailable= '$FemaleBunksAvailable',
 MixedBunksAvailable = '$MixedBunksAvailable',
 RoomsAvailable = '$RoomsAvailable',
 HoursOfOperaion= '$HoursOfOperation',
 ConditionsForUse= '$ConditionsForUse')
WHERE SiteID='$SiteID';

UPDATE foodpantry set HoursOfOperaion= '$HoursOfOperation',
ConditionsForUse = '$ConditionsForUse'
WHERE SiteID='$SiteID';
```

- After update success, a display a confirmation message

Delete service

- When the **Delete** button is clicked, show an alert dialog that says 'Are you sure you want to delete the service "<service type>"?'.
- If the user clicks **Yes**, run the delete statement for the appropriate table**.** If the user clicks **No**, close the dialog and do nothing.

```
DELETE FROM FoodBank WHERE SiteID = $SiteID
DELETE FROM Shelter WHERE SiteID = $SiteID
DELETE FROM SoupKitchen WHERE SiteID = $SiteID
DELETE FROM FoodPantry WHERE SiteID = $SiteID
```

Add Item

Abstract Code

- Run the ***Add Item*** task based on $Username.
- Find the associated $SiteID related to the user.
- Determine if site is within the Food Bank Service table.

```
SELECT * FROM Item WHERE SiteID = $SiteID
```

- If Site ID is not found, the page will display the following text: "Food bank is not found at $SiteName."
- If Site ID is found, a list of items representing food inventory for the associated site will be displayed.
- Entry boxes with labels corresponding to the columns within the Item table will be displayed.
- Once the form is filled out correctly, user will click **Submit.** If valid input, insert a new record in the Items table using the logged in user's $SiteID. Otherwise show a validation message.

```
INSERT INTO Item (ItemID, ItemName, NumberOfUnits, ExpirationDate,
StorageType, SiteID) VALUES ('$ItemID', '$ItemName', '$NumberOfUnits' ,
'$ExpirationDate','$StorageType','$SiteID');
```

Search/Update/Request Item(s)

Abstract Code

- The following filters will appear:
    - Drop-down for site (Individual site names will appear as well as an option for all sites)
    - Date selector for expiration date
    - Storage type drop-down
    - Food/Supply dropdown
    - Drop-downs for sub-categories within Food/Supply
    - Search box for item name/description
    - A button for ***Search***
- Upon clicking the button for ***Search***, a list of items representing food inventory will be displayed based on results from the search criteria:

(Example Query: app will compose appropriate query based on which inputs are filled out)

```
SELECT * FROM Item WHERE Sitename == $SiteName AND ExpirationDate > $ExpirationDate AND StorageType == $StorageType AND FoodSupplyCat == $FoodSupplyCat AND FoodSupplySubCat == $FoodSupplySubCat AND ItemName == $ItemName
```

- For each of the resulting items, the following info and options are displayed:
    - Name of the item
    - Expiration Date
    - An entry box representing quantity of items
    - An ***Update Quantity*** button [IF the item is associated to the current user's site]
    - An entry box representing quantity of items to request [If the item is NOT associated to the current user's site]
    - An ***Request Item*** button [IF the item is NOT associated to the current user's site]
    - A radio button next to each entry within table.
- Find the associated $SiteID related to the user.
- Determine if site is within the Shelter, Soup Kitchen, and Food Pantry tables.

```
SELECT COUNT(SiteID) FROM Shelter WHERE SiteID = $SiteID
SELECT COUNT(SiteID) FROM FoodPantry WHERE SiteID = $SiteID
SELECT COUNT(SiteID) FROM SoupKitchen WHERE SiteID = $SiteID
```

- The user can select an item via the entry button next to the item.

- After user selection of the item, the user can type an integer into the entry box representing an item request. The entry box will be capable of performing simple validation of inputs (e.g. no negative integers, no characters, etc).
- If a valid input is placed in, the ***Item Request*** button is pressed, the Site ID is associated with either a Shelter, Food Pantry, or Soup Kitchen,and the Site ID corresponding to the User is not part of the SiteID for the item requested, text representing a successful request shall be displayed.
- If a valid input is placed in, the ***Item Request*** button is pressed, and the Site ID corresponding to the User is part of the SiteID for the item requested, text representing an unsuccessful request shall be displayed.

```
INSERT INTO Request (Username, ItemID, RequestedQuantity,
FulfilledQuantity, Status) VALUES ('$Username', '$ItemID',
'$RequestedQuantity', '$FulfilledQuantity', '$Status');
```

- After user selection of the item, the user can type an integer into the entry box representing an item update. The entry box will be capable of performing simple validation of inputs (e.g. no negative integers, no characters, etc).
- If a valid input is placed in, the ***Item Update*** button is pressed, and the Site ID corresponding to the User is not part of the SiteID for the item requested, text representing an unsuccessful request shall be displayed.
- If a valid input is placed in, the ***Item Update*** button is pressed, and the $SiteID corresponding to the User is part of the SiteID for the item requested, text representing a successful request shall be displayed. If a quantity is set to zero during the update, it will be removed from the display.

```
UPDATE Item set ItemName='$ItemName',
 NumofUnits= '$NumofUnits',
 ExpDate = '$ExpDate',
 StorageType = '$StorageType',)
WHERE ItemID='$ItemID';
```

Add Client

Abstract Code

- Show a form for new client containing all the relevant fields: *DescriptiveID*, *FirstName, MiddleName, LastName, PhoneNumber (optional)*
- User fills out the form. If all validation passes, insert a new client in the system. Upon success, show a confirmation message.

- If there are any errors on the form, show a validation message

```
INSERT INTO Client (DescriptiveID, FirstName, MiddleName, LastName,
PhoneNumber)
VALUES ('$DescriptiveID', '$FirstName', '$MiddleName', '$LastName',
'$PhoneNumber');
```

Client Search

Abstract Code

- This form contains two search options. “Search By ID” and “Search By Name”
- Option 1: Search By ID. The user will enter a ClientID and click ***Search***

```
SELECT ClientID, DescriptiveID, FirstName, LastName FROMClient WHERE
DescriptiveID like '%$DescriptiveID%';
```

- Option 2: Search By Name: The user will enter the search values and click search. These will be $FirstName and $LastName

```
SELECT ClientID, DescriptiveID, FirstName, LastName FROMClient WHERE
FirstName= '$FirstName' and LastName = '$LastName'
```

- A maximum of 4 results will be shown in a list, with an option to click in for more details to the **Client Detail** form. If no results, then show a message saying so. If more than 4 results, show a message saying the results are truncated and to please enter a more detailed search.

Client Detail

Abstract Code

- This form is to view the details of a client, after they were located through the search. It also displays the complete Client Log and any Waitlists the client is currently on. First load the relevant info from the provided $ClientID:

```
SELECT * FROM Client WHERE ClientID= $ClientID;
```

```
SELECT * FROM ClientLogEntry WHERE ClientID= $ClientID ORDER BY
DateTimeStamp DESC

SELECT * FROM Waitlist WHERE ClientID= $ClientID;
```

- The user can elect to update the current client information. User will update one or more of the fields *DescriptiveID*, *FirstName, MiddleName, LastName, PhoneNumber*, then click ***Save***. The system will update the record

```
UPDATE Client SET DescriptiveID='$DescriptiveID', FirstName='$FirstName',
MiddleName='$MiddleName', LastName='$LastName', PhoneNumber =
'$PhoneNumber' WHERE ClientID = '$ClientID';
```

- The system will do a comparison to analyze which fields are updated. If the DescriptiveID field is updated, create a log entry specifying the old value:

```
INSERT INTO ClientLogEntry (ClientID, DateTimeStamp, SiteName,
Description) VALUES ('$ClientID', '$currentdatetime', '$SiteName',
'DescriptiveID field updated. Old value:$DescriptiveID');
```

- If any of the name fields are updated, create a log entry specifying the old value:

```
INSERT INTO ClientLogEntry (ClientID, DateTimeStamp, SiteName,
Description) VALUES ('$ClientID', '$currentdatetime', '$SiteName','Name
fields updated. Old values: FirstName:$FirstName, MiddleName:
$MiddleName, LastName: $LastName');
```

- This page also provides a function to append custom log entries to the log, such as "Check in". There is a single *LogDescription* for this and a ***Save Log*** button

```
INSERT INTO ClientLogEntry (ClientID, DateTimeStamp, SiteName,
Description) VALUES ('$ClientID', '$currentdatetime', '$SiteName',
'$LogDescription');
```

- This page also provides a button called ***Add to Waitlist***. It is enabled only if the collection of waitlists retrieved earlier does not already contain an entry associated to the current user's $SiteID. Upon clicking, determine the newest (highest) waitlist position for the given site and assign it to a local variable $Rank

```
SELECT COUNT(*) FROM WaitList WHERE ClientID= '$ClientID';
```

```
SELECT MAX(Ranking) FROM WaitList WHERE SiteID = '$SiteID';
```

- Then add the client with current $ClientID to the current site Waitlist

```
INSERT INTO WaitList (ClientID, SiteID, Ranking) VALUES ('$ClientID',
      '$SiteID', '$Rank' + 1);
```

Wait List Report

Abstract Code

- Query the WaitList table for the logged in user's $SiteID and display the results in a grid. If the query returns nothing, show a message saying the current wait list is empty:

```
SELECT * FROM WaitList WHERE SiteID = '$SiteID' ORDER BY Ranking ASC
```

- Each entry has three options: ***Move up, Move down, and Delete.*** With the exceptions that the first entry can't be moved up and the last entry cannot be moved down.
- Upon clicking, the application will calculate the necessary changes to the wait list item $Ranking, and save each one accordingly:
- This is based on the selected client's $ClientID and the logged in user's $SiteID.

```
UPDATE WaitList SET Ranking = $Ranking WHERE ClientID = '$ClientID' AND
SiteID = '$SiteID';
```

- Or Delete if deleted:

```
DELETE FROM WaitList WHERE ClientID = '$ClientID' AND SiteID = '$SiteID';
```

Requests

Outstanding Item Requests

Abstract Code

- Run the **Outstanding Item Requests** task based on $Username.
- Find the associated $SiteID related to the user.
- Determine if site is within the Items table.

```
SELECT COUNT(SiteID) FROM Item WHERE SiteID = '$SiteID';
```

- If Site ID is not found, the page will display the following text: “Food bank inventory empty.”
- If Site ID is found, the system will find the associated items within the Requests relationship table.

```
SELECT Request.ItemID, Request.RequestedQuantity,
Request.FulfilledQuantity, Request.Status, Item.ItemName,
Item.NumofUnits, Item.ExpDate, Item.StorageType, Supplies.Category
FROM Request
  JOIN Item ON Item.ItemID = Request.ItemID
  JOIN Supplies ON Supplies.ItemID = Item.ItemID
WHERE Item.SiteID='$SiteID';
```

```
SELECT Request.ItemID, Request.RequestedQuantity,
Request.FulfilledQuantity, Request.Status, Item.ItemName,
Item.NumofUnits, Item.ExpDate, Item.StorageType, Food.Category
FROM Request
  JOIN Item ON Item.ItemID = Request.ItemID
  JOIN Food ON Food.ItemID = Item.ItemID
WHERE Item.SiteID='$SiteID';
```

- If Item ID associated with the user’s site is not found within the Requests table, the page will display the following text: “No requests for have been made.”
- If Item ID associated with user’s site is not found within the Requests table, the page will display a list of requested items for the site associated with the user.
- The user can click on labels representing storage type, category, sub-category, and quantity of items requested. In doing so, all entries shall be sorted by the column label clicked.
- If the requested quantity found within the Request table for a given Item ID surpasses the number of units found within the Item table, the whole entry will be in red text.
- The fulfilled quantity found for each entry is editable.
- The status for each entry represented as a drop-down showing Pending or Closed.
- After edits are made to fulfilled quantity and status fields, the ***Submit*** button can be pressed.

- Upon pressing of ***Submit***, the number of units for the given Item ID affected will be modified.

```
UPDATE Item SET NumOfUnits='$FulfilledQuantity', WHERE ItemID =
'$ItemID';
UPDATE Request SET FulfilledQuantity='$FulfilledQuantity', SET Status=
'$Status', WHERE ItemID = '$ItemID';
```

Item Requests

Abstract Code

- Run the **Item Requests** task based on $Username.
- Determine if Username is within the Requests relationship table.

```
SELECT * FROM Requests WHERE Username = '$Username';
```

- If Username is not found, the page will display the following text: “Requests for items have not been made.”
- If Username is found, a list of requests for the associated user will be displayed.
- For requests which have not been closed (depicted by the attribute being set to ‘Pending’), a ***Cancel Request*** button shall appear next to the entry.
- If the ***Cancel Request*** button is pressed, the entry shall be deleted from the displayed table.

```
DELETE FROM Requests WHERE ItemID = '$ItemID';
```

Rooms/Bunks Availability Report

Abstract Code

- This form loads and displays the Rooms/Bunks Availability Report
- This report provides a list of all shelters that have available bunks or rooms.
- A shelter with zero availability of rooms,bunks is not shown.
- If no bunks , rooms are available throughout the ASA system , a message is displayed “Sorry, all shelters are at their maximum capacity “ .

```
SELECT Site.SiteID, SiteName,City, 'State', PrimaryContactNumber,
MaleBunksAvailable, FemaleBunksAvailable, MixedBunksAvailable, RoomsAvailable
HoursOfOperation
FROM Site
  JOIN Shelter ON Site.SiteID = Shelter.SiteID
WHERE Shelter.MaleBunksAvailable > 0 OR Shelter.FemaleBunksAvailable > 0 OR
Shelter.MixedBunksAvailable > 0 OR Shelter.RoomsAvailable > 0 ;
```

Meals Remaining Report

Abstract Code

- This form loads and displays the Meals Remaining Report
- This report displays the number of meals remaining in inventory in all food banks in the ASA system .
- The report also displays the type of donations needed to provide more meals(example: Vegetables , or Dairy/eggs or Nuts/Grains)
- The application will calcluate the two values based on the below query, and the formula: Meals Remaining = Minimum of [Vegetable, Nuts/grains/beans, Meat/seafood + Dairy/eggs]
- The lowest of the 3 values = donations most needed

```
SELECT CategoryOfFood, SUM(NumberOfUnits)
FROM Item
  JOIN Food on Item.ItemID = Food.ItemID
WHERE ExpirationDate > NOW()
GROUP BY CategoryOfFood
```