Phase 2 Abstract Code w/SQL

CS 6400 - Spring 2017

Team 010

Table of Contents

Authenticate

Abstract Code

- User enters *username* (‘$UserName’), *password* (‘$Password’) input fields.
- If *username* and *password* fields are both valid, then:

```
SELECT COUNT(*) as ct FROM User
WHERE UserName = ‘$UserName’ AND `Password` = ‘$Password’;
```

    - When ***Enter*** button is clicked:
        - Go to ***Available Services Report*** form
- Else:
    - Go back to ***Login Form*** with error message

View / Add / Edit / Delete Services

Abstract Code

- Run **View Services** task based on $UserName
- Find the current User
- Find the Site the User works for, store as $Site
- Find the Services provided by the Site
- For each Service:
    - Populate *description*, *hours*, *conditions* input fields

```
SELECT DISTINCT sk.ServiceID, sk.Name AS SoupKitchen_Name, sk.Address,
                sk.HoursOfOperation, sk.ConditionsForUse,
                s.ServiceID, s.Name AS Shelter_Name, s.Address, s.HoursOfOperation,
                s.ConditionsForUse,
                fp.ServiceID, fp.Name as FoodPantries_Name, fp.Address,
                fp.HoursOfOperation, fp.ConditionsForUse,
                fb.ServiceID
FROM User u
LEFT JOIN Shelter s on u.SiteID = s.SiteID
LEFT JOIN SoupKitchen sk on u.SiteID = sk.SiteID
LEFT JOIN FoodPantry fp on u.SiteID = fp.SiteID
LEFT JOIN FoodBank fb on u.SiteID = fb.SiteID
WHERE u.UserName =  ‘$UserName’ AND u.SiteID = ‘$Site’;
```

    - If Service is a Food Bank:

        - Show ***Accept Donation*** button
        - Show ***View Outstanding Requests*** button
    - Else:
        - Show ***Register Client*** button
        - Show ***Check In Client*** button
        - Show ***Request Item*** button
        - Show ***Request Status*** button
        - If Service is a Soup Kitchen:
            - Display Soup Kitchen.seats available

```
SELECT sk.SeatsAvailable, sk.SeatCapacity
FROM User u
JOIN SoupKitchen sk on u.SiteID = sk.SiteID
WHERE u.UserName =  '$UserName';
```

            - Show ***Modify Seats Available*** button
        - Else if Service is a Shelter:
            - Display rooms available
            - Show ***Modify Rooms Available*** button
            - Show ***View/Edit Waitlist*** button.
            - Find all Bunk Rooms
            - For each Bunk Room:
                - Display type of room (Male Only, Female Only, Mixed)

```
SELECT br.RoomNumber, br.BunkType, br.BunksAvailable,
br.BunkCapacity
FROM User u
JOIN Shelter s on u.SiteID = s.SiteID
JOIN BunkRoom br on s.ServiceID = br.ServiceID
WHERE u.UserName =  '$UserName';
```

                - Display bunks available; show ***Modify Bunk Count*** button.
    - Show ***Update This Service*** button
    - Show ***Delete This Service*** button
- Show *service type* drop-down list, ***Add Service*** button
- Upon button press, store associated Service as $Service, then:
    - Press ***Accept Donation*** button:
        - Go to ***Accept Donation Form***
    - Press ***View Outstanding Requests*** button:
        - Go to ***Outstanding Requests Report***
    - Press ***Register Client*** button:
        - Go to ***Register Client Form***

- Press ***Check In Client*** button:
  - Go to ***Client Search Form***
- Press ***Request Item*** button:
  - Go to ***Item Search Report***
- Press ***Request Status*** button:
  - Go to ***Request Status Report***
- Press ***Modify Seats Available*** button:
  - Go to ***Seats Available Report*** form.
- Press ***Modify Rooms Available*** button:
  - If Shelter has at least one Family Room:

```
SELECT COUNT(*)
FROM User u
JOIN Shelter s ON u.SiteID = s.SiteID
JOIN FamilyRoom fr ON s.ServiceID = fr.ServiceID
WHERE u.UserName =  '$UserName';
```

    - Go to ***Rooms Available Report*** form.
- Press ***View/Edit Waitlist*** button:
  - If Shelter has at least one Family Room:
    - Go to ***Waitlist Report*** form
- Press ***Modify Bunk Count*** button:
  - If Shelter has at least one Bunk Room:

```
Select COUNT(*)
FROM User u
JOIN Shelter s ON u.SiteID = s.SiteID
JOIN BunkRoom br ON s.ServiceID = br.ServiceID
WHERE u.UserName =  '$UserName';
```

    - Go to ***Bunk Count Report*** form.
- Press ***Update This Service*** button:
  - **Edit Service** task using *description* ($Name, $Address), *hours* ($Hours), *conditions* ($Conditions) input fields

```
UPDATE SoupKitchen
SET Name = '$Name', Address = '$Address', HoursOfOperation = '$Hours',
      ConditionsForUse = '$Conditions'
WHERE SiteID = '$Site';
```

```
UPDATE FoodPantry
SET Name = '$Name', Address = '$Address', HoursOfOperation = '$Hours',
```

```
    ConditionsForUse = '$Conditions'
WHERE SiteID = '$Site';
```

```
UPDATE Shelter
SET Name = '$Name', Address = '$Address', HoursOfOperation = '$Hours',
    ConditionsForUse = '$Conditions'
WHERE SiteID = '$Site';
```

        - **View Services** task

- Press ***Delete This Service*** button:
    - If current Service is last service associated with current Site:
        - Display error message: "Cannot delete last service for this site"
    - Else:
        - Ask for confirmation; if received:
            - **Delete Service** task

```
DELETE FROM SoupKitchen WHERE SiteID = '$Site';
```

```
DELETE FROM FoodBank WHERE SiteID = '$Site';
```

```
DELETE FROM FoodPantry WHERE SiteID = '$Site';
```

```
DELETE FROM Shelter WHERE SiteID = '$Site';
```

            - **View Services** task

- Press ***Add Service*** button:
    - Get new *service type* from input field
    - If current site already has a service of that type:
        - Display error message: "Cannot have two of the same service types"
    - Else:
        - **Add Service** task to make new service, using *service type* input field and default values

```
INSERT INTO SoupKitchen (Name, Address, HoursOfOperation,
ConditionsForUse, SeatCapacity, SeatsAvailable, SiteID)
VALUES ('$Name','$Address','$Hours', '$Conditions', '$Number1',
'$Number2','$Site');
```

```
INSERT INTO FoodBank (SiteID) VALUES ('$Site');
```

```
INSERT INTO FoodPantry (Name, Address, HoursOfOperation,
ConditionsForUse, SiteID)
VALUES ('$Name','$Address','$Hours', '$Conditions','$Site');
```

```
INSERT INTO Shelter (Name, Address, HoursOfOperation,
ConditionsForUse, RoomsAvailable, SiteID)
VALUES ('$Name','$Address','$Hours', '$Conditions', '$Number','$Site');
```

- **View Services** task

Modify Seats Available

Abstract Code

- Find current Soup Kitchen ('$ServiceID').
- **Read Seats Available** task.
- Display seat capacity.

```
SELECT SeatCapacity, SeatsAvailable FROM SoupKitchen WHERE ServiceID='$ServiceID';
```

- Populate *seats available* input field with current value.
- Show ***Update*** button.
- On ***Update*** button press:
    - If *seats available* ('$Number') field is valid (less than capacity):
        - **Update Seats Available** task**.**

```
UPDATE SoupKitchen SET SeatsAvailable= '$Number' WHERE ServiceID='$ServiceID';
```

        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message

Modify Bunk Count

Abstract Code

- Find current Shelter (‘$ServiceID’).
- Find all Bunk Rooms associated with the Shelter.
- For each Bunk Room:
    - **Read Bunks Available** task.

```
SELECT BunksAvailable, RoomNumber FROM BunkRoom
WHERE ServiceID='$ServiceID';
```

    - Display bunk capacity.
    - Populate *bunks available* input field with current value.
- Show ***Update*** button.
- On ***Update*** button press:
    - If input value (‘$Number’) for *bunks available* field is valid (less than capacity and greater than or equal to zero) for all Bunk Rooms:
        - For each Bunk Room (‘$Rooms’):
            - **Update Bunks Available** task**.**

```
UPDATE BunkRoom SET BunksAvailable= '$Number'
WHERE ServiceID='$ServiceID' AND RoomNumber= '$Rooms';
```

        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message.

Modify Rooms Available

Abstract Code

- Find current Shelter (‘$ServiceID’).
- **Read Rooms Available** task.

```
SELECT RoomsAvailable FROM Shelter WHERE ServiceID='$ServiceID';
```

- Populate *rooms available* input field with current value from Shelter.
- Find all FamilyRooms associated with the Shelter.

```
SELECT * FROM FamilyRoom WHERE ServiceID='$ServiceID';
```

- For each FamilyRoom:
    - Populate *room number* and *occupation status* input fields with FamilyRoom.RoomNumber and FamilyRoom.OccupationStatus
- Show ***Update*** button.
- On ***Update*** button press:
    - If input value valid for all *rooms available* ('$Rooms') and *occupation status* ('$Status') fields for each room number ('$Number') (*rooms available* equals number of rooms with *occupation status* = False):
        - **Update Rooms Available** task.

```
UPDATE Shelter SET RoomsAvailable = '$Rooms'
WHERE ServiceID='$ServiceID';
```

        - For each FamilyRoom:
            - **Update Occupation Status** task**.**

```
UPDATE FamilyRoom SET OccupationStatus = '$Status'
WHERE ServiceID='$ServiceID' AND RoomNumber='$Number';
```

        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message.

View / Edit Waitlist

Abstract Code

- Find current Shelter (‘$ServiceID’).
- **View Waitlist** task: Find all Clients in WaitList relationship with Shelter, sorted by WaitList.WaitListPosition

```
SELECT C.ClientID, C.FirstName, C.LastName, W.WaitListPosition
FROM WaitList AS W, Client AS C
WHERE W.ServiceID='$ServiceID' AND W.ClientID = C.ClientID
ORDER BY WaitListPosition;
```

- If no WaitList relationships found:
    - Display “Wait list is empty”
- Else:
    - For each result:
        - Display Client.FirstName, Client.LastName, WaitList.WaitListPosition
        - Store Client.ClientID as ‘$ClientID’, store WaitList.WaitListPosition as ‘$Position’ for current row
        - Show *selection* radio button
    - Show ***Add, Delete, Move Up,*** and ***Move Down*** buttons
    - On ***Add*** button press:
        - Go to ***Client Search Form***
    - If *selection* is set:
        - Retrieve ‘$ClientID’ and ‘$Position’ for selection
        - On ***Delete*** button:
            - Ask for confirmation; if received:
                - Remove Client from WaitList relationship

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

                - Move all other Clients below deleted Client up by one position

```
UPDATE WaitList
SET WaitListPosition = WaitListPosition - 1
WHERE ClientID <> '$ClientID' AND
        WaitListPosition > '$Position';
```

        - On ***Move Up*** button:
            - If Client’s waitlist position is greater than zero: **Update Waitlist** task
                - Find Client just above moving Client; increment their waitlist position

```
UPDATE WaitList
SET WaitListPosition = WaitListPosition + 1
WHERE ClientID <> '$ClientID' AND
        WaitListPosition = '$Position' - 1;
```

                - Decrement Client’s waitlist position

```
UPDATE WaitList
SET WaitListPosition = WaitListPosition - 1
WHERE ClientID = '$ClientID';
```

- Else: Display error message

- On ***Move Down*** button:
  - If Client is not at end of waitlist: **Update Waitlist** task
    - Find Client just below moving Client; decrement their waitlist position

```
UPDATE WaitList
SET WaitListPosition = WaitListPosition - 1
WHERE ClientID <> '$ClientID' AND
        WaitListPosition = '$Position' + 1;
```

    - Increment Client's waitlist position

```
UPDATE WaitList
SET WaitListPosition = WaitListPosition + 1
WHERE ClientID = '$ClientID';
```

  - Else: Display error message

View Beds / Rooms Available

Abstract Code

- For each site, find the number of family rooms that are unoccupied and the number of bunks available of each type

```
SELECT H.Name, H.Address, S.PrimaryPhone, H.HoursOfOperation,
        H.ConditionsForUse, H.RoomsAvailable,
        SUM(CASE WHEN B.BunkType='Male' THEN B.BunksAvailable ELSE 0
            END) AS MaleBunks,
        SUM(CASE WHEN B.BunkType='Female' THEN B.BunksAvailable ELSE 0
            END) AS FemaleBunks,
        SUM(CASE WHEN B.BunkType='Mix' THEN B.BunksAvailable ELSE 0
            END) AS MixedBunks
FROM Shelter AS H
   INNER JOIN Site AS S ON S.SiteID = H.SiteID
```

```
    INNER JOIN BunkRoom AS B ON (H.ServiceID = B.ServiceID)
WHERE (H.RoomsAvailable > 0 OR B.BunksAvailable > 0);
```

- If no rooms or bunks are available (no results from query):
    - Display “Sorry, all shelters are currently at maximum capacity”
- Else:
    - For each result:
        - display Shelter.Name, Shelter.Address, Site.PrimaryPhone, Shelter.HoursOfOperation, Shelter.ConditionsForUse, Shelter.RoomsAvailable, and the summed MaleBunks available, FemaleBunks available, and MixedBunks available from BunkRoom

View Remaining Meals

Abstract Code

- Find food Items available that have not expired yet using NumberOfUnit and ExpirationDate attributes
- Aggregate items into 3 categories:
    - Vegetables
    - Nuts/Grains/Beans
    - Meat/Seafood or Dairy/Eggs
- Calculate number of meals available as the minimum of the count of items in the three categories.

```
SELECT Meal.Counts, Meal.Category
FROM ((SELECT SUM(NumberOfUnit) AS Counts, ItemSubType AS Category
        FROM (SELECT ExpirationDate, NumberOfUnit, 'Meat/Seafood or Dairy/Eggs'
                    AS ItemSubType
                FROM Item
                WHERE ItemType='Food' AND (ItemSubType='Meat/Seafood' OR
                                           ItemSubType='Dairy/Eggs')) AS P
        WHERE P.ExpirationDate >= CURDATE()
        GROUP BY ItemSubType)
       UNION
       (SELECT SUM(NumberOfUnit) AS Counts, ItemSubType AS Category
        FROM Item
        WHERE ItemType='Food' AND ExpirationDate >= CURDATE() AND
                (ItemSubType='Vegetables' OR ItemSubType='Nuts/Grains/Beans')
        GROUP BY ItemSubType)) AS Meal
```

```
ORDER BY Meal.Counts ASC
LIMIT 1;
```

- Display number of meals available (Counts) and Category of the minimum as the type of donations most needed

Add Client

Abstract Code

- Display *first name, last name, phone number, id number*, and *id description* input fields.
- Show ***Add Client*** button.
- On ***Add Client*** button press:
    - Check *first name* ($FirstName), *last name* ($LastName), *id number* ($IDNumber), *phone number* ($PhoneNumber) and *id description* ($IDDescription) fields for valid, non-null entries. If valid:
        - Search for **Existing Client** task: Find Client *id number* ($IDNumber) and *id description* ($IDDescription) that match.

```
SELECT *
FROM Client
WHERE IDNumber = '$IDNumber'
        AND IDDescription = '$IDDescription';
```

        - If existing client found:
            - Display error message
        - Else:
            - Add New Client task: Add new entry to Client table,

```
INSERT INTO Client (IDDescription, IDNumber, FirstName, LastName,
PhoneNumber)
VALUES ('$IDDescription', '$IDNumber', '$FirstName', '$LastName',
'$PhoneNumber');
```

    - Else if not valid:
        - Display error message

Client Search

Abstract Code

- Take *client name* ($FirstName, $LastName) or *IDNumber* ($IDNumber) as an input field input field.

- Run **Client Search** task: find all Clients with either Name or ID Number matching the input string.

```
SELECT COUNT(*)
FROM Client
WHERE IDNumber REGEXP '.*$IDNumber.*'
        OR FirstName REGEXP '.*$FirstName.*'
        OR LastName REGEXP '.*$LastName.*';
```

- If 5 or more results:
    - Prompt user to enter a more unique search item
- Else if number of results is between 1 and 5:
    - For each result:
        - Display Client.Full Name, Client.ID Number and Description

```
SELECT FirstName, LastName, IDNumber, IDDescription
FROM Client
WHERE IDNumber REGEXP '.*$IDNumber.*'
        OR FirstName REGEXP '.*$FirstName.*'
        OR LastName REGEXP '.*$LastName.*';
```

        - Show ***Select*** button
- Else:
    - Display “No matching results” message.
- On ***Select*** button press:
    - Go to ***Client Report***, pass selected client as $ClientID

View / Edit Client Information

Abstract Code

- Find current Client from $ClientID.
- **View Client** task
- For each Client:
    - Populate *first name*, *last name, phone number, id number,* and *id description* input fields from Client.First Name, Client.Last Name, Client.Phone Number, Client.ID Number, and Client.ID Description

```
SELECT FirstName, LastName, IDNumber, IDDescription
FROM Client
WHERE ClientID = '$ClientID';
```

- **View Waitlist** task: Find any Waitlist requests associated with the Client, and the Shelter associated with the Waitlist request.
- For each Waitlist request:
  - Display associated Shelter.Name, Waitlist.Waiting List Position

```
SELECT Shelter.Name, Waitlist.WaitListPosition
FROM WaitList INNER JOIN Shelter
        ON Waitlist.ServiceID = Shelter.ServiceID
WHERE Waitlist.ClientID = '$ClientID';
```

- **View Log** task: Find all Log entries associated with the Client
- For each Log entry:
  - Display Log.DateTime, Log.SiteName, Log.ServiceDescription, Log.Notes

```
SELECT Log.LogDateTime, Log.SiteName, Log.ServiceDescription,
        Log.Notes
FROM Log
WHERE Log.ClientID = '$ClientID';
```

- Show ***Update*** button
- Show ***Check In*** button
- On ***Update*** button press:
  - If input values valid for all input fields (phone number may be NULL):
    - **Add Log Entry** task with existing values from Client table, current date/time ($LogDateTime), current SiteName ($SiteName), *service description* for ($ServiceDescription), and notes ($Notes)

```
INSERT INTO Log (ClientID, LogDateTime, SiteName,
ServiceDescription, Notes)
VALUES ('$ClientID', '$LogDateTime', '$SiteName',
        '$ServiceDescription', '$Notes');
```

    - **Update Client** task with new values from input fields
    - **View Client** task
  - Else:
    - Display error message.
- On ***Check In*** button press:

- Go to ***Client Check-In Form***

Check In Client

Abstract Code

- Find current Client from $ClientID
- Find current Site from $SiteID (set on login)
- **Check for Services** task with current Site

```
SELECT
  Shelter.SiteID IS NOT NULL AS provides_shelter,
  SoupKitchen.SiteID IS NOT NULL AS provides_soup_kitchen,
  FoodPantry.SiteID IS NOT NULL AS provides_food_pantry
FROM Site
  LEFT JOIN Shelter
  ON Site.SiteID = Shelter.SiteID
  LEFT JOIN SoupKitchen
  ON Site.SiteID = SoupKitchen.SiteID
  LEFT JOIN FoodPantry
  ON Site.SiteID = FoodPantry.SiteID
WHERE Site.SiteID = '$SiteID';
```

- For each available service:
  - Show *service log* input field; populate with Site and Service information
  - Show ***Add Service Log*** button
  - If service is a Shelter that has at least one Family Room:
    - Show ***Add to Room Waitlist*** button
  - If service is a Shelter that has at least one Bunk Room:
    - Show ***Modify Bunk Count*** button
- On ***Add Service Log*** press:
  - Prepend timestamp to *service log* input field value
  - Store value as log: **Add Client Service Log Entry** task
- On ***Add to Room Waitlist*** button:
  - If current Client is not on Waitlist for current Site: **Add Client to Room Waitlist** task
    - Find last waitlist position $last
    - Add Client to waitlist with position $last + 1

```
INSERT INTO WaitList
SELECT
  ServiceID,
```

```
 '$ClientID' AS ClientID
 (SELECT
   MAX(WaitListPosition) + 1 FROM WaitList) AS WaitListPosition
FROM Shelter
WHERE Shelter.SiteID = '$SiteID';
```

        - Else:
            - Display error message
- On ***Modify Bunk Count*** button:
    - Go to ***Bunk Count Report***

Add Inventory

Abstract Code

- Create the following input fields relating to table Item:
    - *name ($Name)*
    - *number of unit ($NumberOfUnit)*
    - drop-down list specifying *unit type* ($UnitType) of Bag, Box, Carton, or Others
    - *expiration date ($ExpirationDate)*, populated with 01/01/9999 for default
    - *storage type ($StorageType)* drop-down list (Dry Good, Refrigerated, Frozen)
    - *item type ($ItemType)* drop-down list (Food, Supplies)
    - *item sub type* ($ItemSubType) drop-down list (Vegetables, Nuts/Grains/Beans, Meat/Seafood, Dairy/Eggs, Sauce/Condiment/Seasoning, Juice/Drink, Personal Hygiene, Clothing, Shelter, Other)
- Show ***Add Inventory*** button
- On ***Add Inventory*** button press:
    - If all fields valid:
        - **Get User Name**: get user name ($UserName) from the session of the current user logged onto system
        - **Get FoodBank's Service ID:** get service ID ($ServiceID) from joining user name ($UserName) to food bank using site ID

```
SELECT FoodBank.ServiceID
FROM User
INNER JOIN FoodBank
ON User.SiteID = FoodBank.SiteID
WHERE User.UserName = '$UserName';
```

- **Check for Matching Item**: look to see if input item already exist by finding an Item.ItemID (store in $ItemID) that matches *name* and *expiration date* ($ExpirationDate) from the input fields

```
SELECT Item.ItemID
FROM Item
WHERE Item.Name LIKE '%' + '$Name' + '%'
AND Item.ExpirationDate = '$ExpirationDate';
```

- If existing item found:
  - **Edit Existing Item**: Add *number of unit* to Item.NumberOfUnit

```
UPDATE Item
SET Item.NumberOfUnit = Item.NumberOfUnit +
 '$NumberOfUnit'
WHERE Item.ItemID = '$ItemID';
```

- Else:
  - **Add Item to Inventory**: add new item to Item table

```
INSERT INTO Item (ItemType, ItemSubType, Name,
    NumberOfUnit, UnitType, ExpirationDate, StorageType,
    ServiceID)
VALUES ('$ItemType', '$ItemSubType', '$Name',
    '$NumberOfUnit', '$UnitType', '$ExpirationDate',
    '$StorageType', '$ServiceID';
```

Search Item

Abstract Code

- **Retrieve all the Site names and IDs with food bank**: get Site.Name into array $SiteNames and related Site.ID into parallel array $SiteIDs (i.e. first site name in $SiteNames has its site ID in the first position of $SiteIDs, second site name in $SiteNames has its site ID in the second position of $SiteIDs, etc.)

```
SELECT Site.Name, Site.SiteID
FROM Site
INNER JOIN FoodBank
ON Site.SiteID = FoodBank.SiteID;
```

- Display input fields for:
    - *food bank* drop-down list (populated with $SiteNames; selected value stored in $SiteName and store related site ID in $SiteID)
    - *item type* ($ItemType) drop-down list (All, Food, Supplies)
    - *storage type* ($StorageType) drop-down list (All, Dry Good, Refrigerated, Frozen)
    - *item sub type* ($ItemSubType) drop-down list (All, Vegetables, Nuts/Grains/Beans, Meat/Seafood, Dairy/Eggs, Sauce/Condiment/Seasoning, Juice/Drink, Personal Hygiene, Clothing, Shelter, Other)
    - *expiration date after* ($ExpirationDate) calendar selection widget
    - *name* ($Name)
- Display ***Search*** button.
- On ***Search*** button press:
    - Get service ID ($ServiceID) for $SiteName's food bank by using related $SiteID:

```
SELECT FoodBank.ServiceID
FROM FoodBank
WHERE FoodBank.SiteID = '$SiteID';
```

    - Set $sqlStart as 'SELECT * FROM Item '
    - For each input field that is not NULL, add to $whereSQL, which represents the **WHERE** clause, the related Item's column name filtered by the corresponding input field value
    - Look to see if any matches in Item by querying $sqlStart + $whereSQL
        - Example: if user only makes selections for item type and item sub type and leaves all other fields NULL, then $whereSQL will be set as "WHERE Item.ItemType = '$ItemType' AND Item.ItemSubType = '$ItemSubType'" so that the query becomes:

```
SELECT *
FROM Item
WHERE Item.ItemType = '$ItemType'
AND Item.ItemSubType = '$ItemSubType';
```

        If all fields are filled, the query becomes:

```
SELECT *
FROM Item
WHERE Item.ItemType = '$ItemType'
AND Item.StorageType = '$StorageType'
AND Item.ItemSubType = '$ItemSubType'
AND Item.ExpirationDate = '$ExpirationDate'
AND Item.Name Like '%' + '$Name' + '%'
AND Item.ServiceID = '$ServiceID';
```

    - If any matching Items found:
        - Pass results as $Items to ***Available Items Report***
    - Else:
        - Display message: “No matching Items found.”

View / Request / Edit / Delete Item

Abstract Code

- Find the site ID ($SiteID) of the current user ($UserName) set on login session variable

```
SELECT User.SiteID
FROM User
WHERE User.UserName = '$UserName';
```

- Find any Food Bank service IDs associated with current user’s $SiteID, store as $ServiceID

```
SELECT FoodBank.ServiceID
FROM FoodBank
WHERE FoodBank.SiteID = '$SiteID';
```

- **View Items** task:
    - Find all items in Item:

```
SELECT *
FROM Item
WHERE Item.NumberOfUnit > 0;
```

    - For each item in Item:
        - Store Item.ItemID in $ItemID
        - Display in a table format the non-editable fields from Item.ItemID, Item.Name, Item.ItemType, Item.ItemSubType, Item.ExpirationDate, Item.StorageType, Item.UnitType, Item.NumberOfUnit
        - Display editable input text field for *request unit number* ($RequestUnitNumber). Default value equals 0.

- Display ***Update*** button
- On ***Update*** button press:

- Find each item ($EditItem) in Item for which Food Bank is the same as $ServiceID, and retrieve the new number from the display table ($EditNumber)

```
SELECT *
FROM Item
WHERE Item.Service = '$ServiceID' AND Item.NumberOfUnit > 0;
```

- Find each Item's Item.ID ($RequestItemID) for which *request unit number* ($RequestUnitNumber) input field is non-zero

```
SELECT *
FROM Item
WHERE Item.Service = '$ServiceID'
AND Item.ItemID = '$RequestItemID';
```

- Display confirmation message listing each $EditItem matching to $RequestItemID. Show Item.Name, Item.ItemType, Item.ItemSubType, Item.NumberOfUnit, and $RequestUnitNumber.

- If confirmation received:
  - Run **Edit Item** task to update each $EditItemID's Item.NumberOfUnit from *number of item* ($EditNumber) input field. If the new number is less than zero, an error should be displayed. Otherwise:

```
UPDATE Item
SET Item.NumberOfUnit = '$EditNumber'
WHERE Item.ItemID = '$EditItemID';
```

  - Run **Request Item** to make new Request association between each $RequestItem and current User

```
INSERT INTO Requests (UserName, ItemID, CountRequested)
VALUES ($UserName, $RequestItemID, $RequestUnitNumber);
```

- Re-run **View Items** task to refresh table

View / Fill Item Request

Abstract Code

- **Find Food Bank's service ID**: get service ID ($ServiceID) by using the site ID of the current user ($UserName) set on login

```
SELECT FoodBank.ServiceID
FROM User
INNER JOIN FoodBank
ON User.SiteID = FoodBank.SiteID
WHERE User.UserName = '$UserName';
```

- **Read Requests** task: find all Requests associated with the current Food Bank by using user name.

```
SELECT *
FROM Requests
WHERE Requests.UserName = '$UserName';
```

- Display sortable columns: Name, Storage Type, Category, Sub-Category, Number Available, and Number Requested
- For each request:
    - Store Requests.ItemID in $ItemID, Requests.CountRequested in $CountRequested, Request.Status in $Status, Requests.RequestDateTime in $RequestDateTime
    - Find the item associated with the request's $ItemID

```
SELECT *
FROM Item
WHERE Item.ItemID = '$ItemID';
```

    - Display the Item.Name, Item.StorageType, Item.ItemType, Item.ItemSubType, Item.NumberOfUnit, and $CountRequested.
    - Show a *number provided* input field and a *status* drop-down list (Pending, Filled, Partially Filled, Unable to Fill). Populate *status* with $Status.
    - Find any other existing Requests associated with the same Item.

```
SELECT COUNT(*) AS CountTimesItemRequested,
```

```
SUM(Requests.CountRequested) AS TotalRequestedQty
FROM Requests
WHERE Requests.ItemID = '$ItemID';
```

- If the sum of these Requests.CountRequested is higher than the Item.NumberOfUnit:
  - Highlight current request. Requests.CountRequested in red

- Show ***Update*** button
- On ***Update*** button press:
  - If all fields are valid (number provided less than or equal to the number available, status is not pending):
    - **Fill Requests / Edit Items** task
    - For each request:
      - Update Requests.Status with $UpdatedStatus and set Requests.CountRequested with $UpdatedCount from *status* and *number provided* fields for the current user ($UserName) and by matching to earlier found request date time ($RequestDateTime) and item ID ($ItemID)

```
UPDATE Requests
SET Requests.Status = '$UpdatedStatus',
Requests.CountRequested = '$UpdatedCount'
WHERE Requests.UserName = '$UserName'
AND Requests.ItemID = '$ItemID'
AND Requests.RequestDateTime = '$RequestDateTime';
```

      - Using the Item ($ItemID) associated with the request. Subtract the number provided from the Item.NumberOfUnit..

```
UPDATE Item
SET Item.NumberOfUnit = Item.NumberOfUnit -
'$UpdatedCount'
WHERE Item.ItemID = '$ItemID';
```

    - **Read Requests** task to re-display Request list
  - Else:
    - Display error message

View / Cancel Item Request

Abstract Code

- Find current User from $UserName, set on login.
- Find all Requests with status pending.

```
SELECT *
FROM Requests
WHERE Requests.UserName = '$UserName'
AND Requests.Status = 'Pending';
```

- **View Requests** task:
- For each Request.ItemID:
    - Store Requests.ItemID in $ItemID, Requests.CountRequested in $CountRequested, Requests.CountProvided in $CountProvided, Request.Status in $Status, Requests.RequestDateTime in $RequestDateTime
    - Find the Item associated with the request.

```
SELECT *
FROM Item
WHERE Item.ItemID = '$ItemID';
```

    - Display the Item.Name, Item.StorageType, Item.ItemType, Item.ItemSubType, Item.NumberOfUnit, and $CountRequested, $CountProvided, $RequestDateTime, $Status.
    - Show a *cancel* check box.
- Show a ***Cancel*** button.
- On ***Cancel*** button press:
    - **Cancel Requests** task:
    - Find each Request (with with a *cancel* check box selected)
    - Ask user for confirmation. If received: delete the Request from the table.

```
DELETE
FROM Request
WHERE Request.UserName = '$UserName'
AND Request.ItemID = '$ItemID'
AND Request.RequestDateTime = '$RequestDateTime';
```

    - **View Requests** to refresh table.