Table of Contents

Login

Abstract Code

- User populates the *username* (`$Username`), *password* (`$Password`) input fields.
- When ***Login*** button is clicked:
  If the *username* or *password* field is empty:
    - Stay in **Login** form, with the message “Username and password are required. Please try again”
  - If both *username* and *password* fields are populated:
    - Query the User table with the *username* value.

```
SELECT password
 FROM User WHERE username = `$Username`;
```

    - If User record is NOT found:
      - Go back to **Login** form, with error message “Invalid login. Please try again.”
    - If User record is found but User.password != `$Password`:
      - Go back to **Login** form, with error message “Invalid login. Please try again.”
    - Else
      - Store login information as session variable `$UserData`.
      - Go to **User Home** form.

Client Search

Abstract Code

- User populates the *clientFirstName* (`$ClientFirstName`) and/or *clientLastName* (`$ClientLastName`) and/or *description* (`$Description`) input fields.
- When ***Search*** button is clicked:
  - If both *clientFirstName* and *clientLastName* and *description* are empty:
    - Stay in **Client Search** form, with the message “Client first name or last name or description must be populated, please try again.”
  - If *clientFirstName* and/or *clientLastName* and/or *description* are populated:
    - Lookup the Client table based on the search criteria(s) provided.

```
SELECT clientId, firstName, lastName, description FROM Client
WHERE firstName = IFNULL(`$ClientFirstName`,firstName) AND
lastName = IFNULL(`$ClientLastName`,lastName) AND description =
IFNULL(`$Description`,description);
```

    - If no Client record is found:

- Go back to **Client Search** form, with the message "No Client found, please try again."
- If 5 or more Client records are found:
  - Go back to **Client Search** form, with the message "Too many Clients found, please narrow search criteria(s)."
- Else
  - Retrieve the matching clients data and store as session variable \`$ClientSearchResults\`.
  - Go to **Client Search** form to display the search results

- When ***Detail*** button is clicked (note that ***Detail*** button is enabled only when a Client record has been selected):
  - Retrieve the Client record from the \`$ClientIdSelected\` session variable.

```
SELECT clientId, firstName, lastName, description FROM Client WHERE
clientId = `$ClientIdSelected`;
```

  - Lookup all modifications to Client records.

```
SELECT clientId, dateTime, fieldModified, previousValue FROM ClientLog
WHERE clientId = `$ClientIdSelected`;
```

  - Lookup all services provided to Client.

```
SELECT clientId, siteId, dateTime, description, note FROM ClientService
WHERE clientId = `$ClientIdSelected`;
```

  - Store modifications and services provided in the \`$ClientData\` session variables.
  - Go to **Client Report** form.
- When ***Enroll*** button is clicked:
  - Go to **Enroll Client** form. No data is required beforehand to enroll a new client.

Enroll Client

Abstract Code

- User enters *clientFirstName* (\`$ClientFirstName), *clientLastName* (\`$ClientLastName), *description* (\`$Description), and/or *phoneNumber* (\`$PhoneNumber\`) input fields.
- When the **Enroll Client** button is clicked:
  - If *clientFirstName* or *clientLastName* or *description* field is empty:
    - Stay in **Enroll Client** form, with a message of both client first name, last name, and description must be provided
  - If both *clientfirstName*, *clientLastName*, and *description* are provided:

- Lookup *clientFirstName*, *clientLastName*, and *description* values on the Client table

```
SELECT clientId, firstName, lastName, description FROM Client
WHERE firstName = IFNULL(`$ClientFirstName`,firstName) AND
lastName = IFNULL(`$ClientLastName`,lastName) AND description =
IFNULL(`$Description`,description);
```

- If one or more Client records exist with *clientFirstName*, *clientLastName*, and *description* values:
  - Go back to **Enroll Client** form, with a message of Client already exists.
- Else
  - Create a new Client record with *clientFirstName, clientLastName, description,* and *phoneNumber*.

```
INSERT INTO Client (clientId, firstName, lastName, description,
phone) VALUE (NULL, `$ClientFirstName`,`$ClientLastName`,
`$Description`,`$PhoneNumber`;
```

  - Store the Client information as a session variable `$ClientData`.
  - Go to **Client Report** form.

Modify Client

Abstract Code

- User modifies *clientFirstName* (`$ClientFirstName) and/or *clientLastName* (`$ClientLastName) and/or *description* (`$Description`) and/or *phoneNumber* (`$PhoneNumber`) fields in **Client Report** form.
- When the **Update** button is clicked:
  - If *clientFirstName* or *clientLastName* or *description* field is empty:
    - Stay in **Client Report** form, with a message of both client first name, last name, and description must not be empty.
  - If *clientFirstName*, *clientLastName,* and *description* are populated:
    - If *clientFirstName* was updated:

```
INSERT INTO ClientLog (clientId, userId, dateTime, fieldModified,
previousValue) VALUES
(`$ClientId`,`$UserId`,sysdate(),`firstname`,`$ClientFirstNamePrevious);
```

- If *clientLastName* was updated:

```
INSERT INTO ClientLog (clientId, userId, dateTime, fieldModified, previousValue) VALUES
(`$ClientId`,`$UserId`,sysdate(),`lastname`,`$ClientLastNamePrevious);
```

- If *description* was updated:

```
INSERT INTO ClientLog (clientId, userId, dateTime, fieldModified, previousValue) VALUES
(`$ClientId`,`$UserId`,sysdate(),`description`,`$DescriptionPrevious);
```

- If *phoneNumber* was updated:

```
INSERT INTO ClientLog (clientId, userId, dateTime, fieldModified, previousValue) VALUES
(`$ClientId`,`$UserId`,sysdate(),`phonenumber`,`$PhoneNumberPrevious);
```

- If *clientFirstName*, *clientLastName, description*, or *phoneNumber* was updated:

```
UPDATE Client SET firstName = `$ClientFirstName`, lastName =
`$ClientLastName`, description =`$Description` , phoneNumber =
`$PhoneNumber` WHERE clientId = `$ClientId`;
```

- If update was successful:
  - Lookup Client record using the *clientId* from the `$ClientData` session variable.

    ```
    SELECT clientId, firstName, lastName, description,
    phoneNumber FROM Client WHERE clientId = `$ClientId`;
    ```

  - Store Client record into the `$ClientData` session variable.
  - Go back to **Client Report** form, with the message Client data updated successful.
- If update was successful:
  - Go back to **Client Report** form, with the message “Unable to save Client data, please try again.”

Check-In Client

Abstract Code

- User populates *datetime* (`$DateTime) and *description* (`$Description`) and/or *note* (`$Note`) fields in **Check-In Client** form.
- When the **Save** button is clicked:
  - If *datetime* or *description* field is empty:
    - Stay in **Check-In Client** form, with the message "Date/time and description are required, please try again.".
  - If both *datetime* and *description* are populated:
    - Insert into ClientService table with the *datetime*, *description,* and *note* values using *clientId* and *site* values stored session variables.

      ```
      INSERT INTO ClientService (clientId, siteId, userId, dateTime, description, note) VALUES (`$ClientId`,`$SiteId`,`$UserId`,`$DateTime`,`Description`,`$Note);
      ```

    - If update was successful:
      - Lookup Client record using the *clientId* from session variable.

        ```
        SELECT clientId, firstName, lastName, description, phoneNumber FROM Client WHERE clientId = `$ClientId`;
        ```

      - Lookup all modifications to Client records.

        ```
        SELECT clientId, dateTime, fieldModified, previousValue FROM ClientLog WHERE clientId = `$ClientId`;
        ```

      - Lookup all services provided to Client.

        ```
        SELECT clientId, siteId, dateTime, description, note FROM ClientService WHERE clientId = `$ClientId`;
        ```

      - Store Client data, modifications and services provided in the `$ClientData` session variables.
      - Go to **Client Report** form.
    - If update was successful:
      - Go back to **Check-In Client** form, with the message "Unable to Check-In, please try again."
- When the **Cancel** button is clicked:

- Go back to **Client Report** form.

Reset Bunk Count

Abstract Code

- View site information from **Site Form**.
- While no buttons are pushed, do nothing.
- When ***Update bunk count*** button is pushed:
  - Find the available bunk information(`$Male, $Female, $Mixed`) in Shelter table
  - Display the retrieved bunk count(`$Male, $Female, $Mixed`)
  - If ***Save*** button is pushed:
    - Go to **Site Form** and save all the information
    - Go back to site information page
  - If ***Cancel*** button is pushed:
    - Go back to site information page

```
SELECT Shelter.Male, Shelter.Female, Shelter.Mixed
FROM Shelter
INNER JOIN ClientService
ON ClientService.FacilityId = Shelter.FacilityId
WHERE ClientService.FacilityName = `$FacilityName`
```

Report Available Bunks

Abstract Code

- The information about available bunks will be updated by system at a daily base.
- Once the updating starts, first inner join the bunk information from Shelter table and detailed shelter information from ClientService table, and store the result into BunkInfoResult
  - If zero availability of bunks in this shelter:
    - Skip the shelter

```
SELECT
   Shelter.FacilityId, Shelter.BunkCountMale, Shelter.BunkCountFemale,
   Shelter.BunkCountMixed, ClientService.FacilityName, ClientService.EligibilityCondition,
   ClientService.HoursOfOperation
FROM Shelter INNER JOIN ClientService
ON Shelter.FacilityId = ClientService.FacilityId
WHERE Shelter.BunkCountMale > 0
OR Shelter.BunkCountFemale >0
OR Shelter.BunkCountMixed >0;
```

- Join SiteId from SiteToService table

```
SELECT
   Shelter.FacilityId, Shelter.BunkCountMale, Shelter.BunkCountFemale,
   Shelter.BunkCountMixed, ClientService.FacilityName, ClientService.EligibilityCondition,
   ClientService.HoursOfOperation, SiteToService.SiteId
FROM BunkInfoResult
LEFT JOIN SiteToService
ON Shelter.FacilityId = SiteToService.FacilityId;
```

- store the result into BunkInfoResult
- join site information from Site table

```
SELECT
   Shelter.FacilityId, Shelter.BunkCountMale, Shelter.BunkCountFemale,
   Shelter.BunkCountMixed, ClientService.FacilityName, ClientService.EligibilityCondition,
   ClientService.HoursOfOperation, Site.LocationStreetAddress, Site.LocationCity,
   Site.LocationState, Site.LocationZipCode, Site.phone
FROM BunkInfoResult
LEFT JOIN Site
ON SiteToService.SiteId = Site.SiteId;
```

  - store the result in **Bunk Report** Form

Report Meals Remaining

Abstract Code

- Once the updating starts, first left join the FoodBankToItem table and food category information from Food table, and store the result into ReportMealResult.

```
SELECT
   FoodBankToItem.FacilityId,
   FoodBankToItem.ItemId,
   FoodBankToItem.AvailableQuantity,
   Food.ItemId,
   Food.FoodCategory
FROM FoodBankToItem
LEFT JOIN Food
ON FoodBankToItem.ItemId = Food.ItemId;
```

- Sum up all the numbers of food in all the food banks under different FoodCategory in ReportMealResult, and save in CategoryResult

```
SELECT
    FoodCategory,
SUM (AvailableQuantity)
FROM ReportMealResult
GROUP BY FoodCategory ;
```

```
SELECT AvailableQuantity into @Veg FROM CategoryResult
WHERE FoodCategory = 'Vegetable';
SELECT AvailableQuantity into @Carb FROM CategoryResult
WHERE FoodCategory = 'Nuts/grains/beans';
SELECT AvailableQuantity into @Protein1 FROM CategoryResult
WHERE FoodCategory = 'Meat/seafood';
SELECT AvailableQuantity into @Protein2 FROM CategoryResult
WHERE FoodCategory = 'Dairy/eggs';
```

- Sum up total number of(`$Protein1`)Meat/seafood and (`$Protein2`)Dairy/eggs as one category(`$Protein`)
- Compare the total numbers of three food category Vegetable(`$Veg`), Nuts/grains/beans(`$Carb`), and Meat/seafood OR Dairy/eggs(`$Protein`) and find the minimum of the three
- Store and display the minimum number as the number of the meals remaining in inventory in all food banks in the ASA system
- Store and display the food category which has the minimum number as type of donations are most needed to provide more meals

Modify Service

Abstract Code

- View site information from **Site Form**. Populate service list dropdowns.
- While no buttons are pushed, do nothing.
- When ***Add*** button is pushed:
    - If the selected service matches any value of ServiceList(`$ServiceList`) of the site:
        - Error message of existing service
    - Else:
        - Insert the service into ServiceList (`$ServiceList`) of the site
        - Display the **Service-Info** form with the selected service
        - If ***Save*** button is pushed:
            - Go to Service table and save all the information
            - Go back to site information page
        - If ***Cancel*** button is pushed:
            - Go back to site information page

```
IF EXISTS (SELECT * FROM SiteToService WHERE ServiceList =`$ServiceList`)
BEGIN
        --Error Message
END
ELSE BEGIN
        IF `$ServiceList` == `FoodBank` then
          INSERT INTO Service (facilityId) VALUES (NULL);
          Save last inserted value into $FacilityId;
          INSERT INTO ClilentService
(facilityId,facilityName,eligibilityCondition,hoursOfOperation) VALUES
(`$FacilityId`,`$FacilityName`,`$EligibilityCondition`,`$HoursOfOperation`);
          INSERT INTO FoodBank (facilityId) VALUES (`$FacilityId`);
          INSERT INTO SiteToService(siteId, facilityId) VALUES (`$SiteId`,`$FacilityId`);
        IF `$ServiceList` == `FoodPantry` then
          INSERT INTO Service (facilityId) VALUES (NULL);
          Save last inserted value into $FacilityId;
          INSERT INTO ClientService
(facilityId,facilityName,eligibilityCondition,hoursOfOperation) VALUES
(`$FacilityId`,`$FacilityName`,`$EligibilityCondition`,`$HoursOfOperation`);
          INSERT INTO FoodPantry (facilityId) VALUES (`$FacilityId`);
          INSERT INTO SiteToService(siteId, facilityId) VALUES (`$SiteId`,`$FacilityId`);
        IF `$ServiceList` == `Shelter` then
          INSERT INTO Service (facilityId) VALUES (NULL);
          Save last inserted value into $FacilityId;
          INSERT INTO ClientService
(facilityId,facilityName,eligibilityCondition,hoursOfOperation) VALUES
```

```
(`$FacilityId`,`$FacilityName`,`$EligibilityCondition`,`$HoursOfOperation`);
          INSERT INTO Shelter
(facilityId,bunkType,bunkCountMale,bunkCountFemale,bunkCountMixed) VALUES
(`$FacilityId`,`$BunkType`,`$BunkCountMale`,`$BunkCountFemale`,`$BunkCountMixed`);
          INSERT INTO SiteToService(siteId, facilityId) VALUES (`$SiteId`,`$FacilityId`);
        IF `$ServiceList` == `SoupKitchen` then
          INSERT INTO Service (facilityId) VALUES (NULL);
          Save last inserted value into $FacilityId;
          INSERT INTO ClientService
(facilityId,facilityName,eligibilityCondition,hoursOfOperation) VALUES
(`$FacilityId`,`$FacilityName`,`$EligibilityCondition`,`$HoursOfOperation`);
          INSERT INTO SoupKitchen (facilityId,seatCount) VALUES
(`$FacilityId`,`$SeatCount`);
          INSERT INTO SiteToService(siteId, facilityId) VALUES (`$SiteId`,`$FacilityId`);
END
```

- When ***Edit*** button is pushed:
  - Display the original **Service-Info** form retrieved from Service table for editing
  - If ***Save*** button is pushed:
    - Go to Service table and save all the information
      - If the service is a shelter.

```
IF `$ServiceList` == `Shelter` THEN

UPDATE ClientService SET facilityID = `$facilityID`, facilityName =
`$facilityName`, eligibilityCondition =`$eligibilityCondition` , hoursOfOperation =
`$hoursOfOperation`;

UPDATE Shelter SET facilityID = `$facilityID`, bunkType = ‘$bunkType’,
maleBunkCount = ‘$maleBunkCount, femaleBunkCount = ‘$femaleBunkCount’,
mixedBunkCount = ‘$mixedBunkCount’

IF `$ServiceList` == `FoodPantry` THEN
UPDATE ClientService SET facilityID = `$facilityID`, facilityName =
`$facilityName`, eligibilityCondition =`$eligibilityCondition` , hoursOfOperation =
`$hoursOfOperation`;

IF `$ServiceList` == `SoupKitchen` THEN
UPDATE ClientService SET facilityID = `$facilityID`, facilityName =
`$facilityName`, eligibilityCondition =`$eligibilityCondition` , hoursOfOperation =
`$hoursOfOperation`;

UPDATE SoupKitchen SET facilityID = `$facilityID`, seatCount = ‘$seatCount’;
```

    - Go back to site information page

- If ***Cancel*** button is pushed:
  - Go back to site information page

- When ***Delete*** button is pushed:
  - If number of element of ServiceList (`$ServiceList`) equals to one:
    - Error message of one service remaining
  - Else:
    - Message of confirmation for service deleting
    - If ***Save*** button is pushed:
    -

```
IF `$ServiceList` == `FoodBank` then
DELETE FROM FoodBank
WHERE SiteId = `$SiteId`
AND FacilityId = `$FacilityId`;

IF `$ServiceList` == `Shelter` then
DELETE FROM Shelter
WHERE SiteId = `$SiteId`
AND FacilityId = `$FacilityId`;

IF `$ServiceList` == `SoupKitchen` then
DELETE FROM SoupKitchen
WHERE SiteId = `$SiteId`
AND FacilityId = `$FacilityId`;

IF `$ServiceList` == `FoodPantry` then
DELETE FROM FoodPantry
WHERE SiteId = `$SiteId`
AND FacilityId = `$FacilityId`;
```

- If ***Cancel*** button is pushed:
- Go back to site information page

Add Item

Abstract Code

- Click ***Add Item*** button from **Item Add** form
-

```
INSERT INTO Item (Name, StorageType, ItemType, ExpirationDate)
VALUE
```

```
(`$Name`, `$StorageType`, `$ItemType`, `$ExpirationDate`);
```

Request Item

Abstract Code

- Click ***Request Item*** button from **Item Report** form. Jump to **Request Item** task.
- Add a request record in Request Table that stores the source (Food Bank), user and other request attributes.

```
INSERT INTO Request (RequestId, ItemId, Status, QuantityRequested,
QuantityFulfilled)
VALUE
(`$RequestId`, `$ItemId`, `$Status`, `$QuantityRequested`,
`$QuantityFulfilled`);
```

Item Search

Abstract Code

- Click ***Search Item*** button from **Item Search** form. Jump to **Search Item** task.
- User specify food bank (`$foodBank`) they want to search items from. It can be wildcard.
  -

```
SELECT
   Item.itemID, Item.name, Item.storageType, Item.expirationDate,
   FoodBankToItem.FacilityName, FoodBankToItem.availableQuantity
FROM FoodBankToItem INNER JOIN Item
ON FoodBankToItem.itemID = Item.itemID
WHERE FoodBankToItem.availableQuantity > 0;
```

- The system will do a join on table FoodBankToItem and Item as above, and store the result into ItemSearchResult for the following queries on item params.
- User specify filters of expiration date (`$expirationDate`), storage type (`$storageType`), Food/Supply (`$foodSupply`), individual category (`$individualCategory`) or keyword of item name/ description (`$name`). Based on what user specify, look up all Food Bank tables to return a list of items.
  - If user specifies filter of name (`$name`)
    -

```
SELECT itemID, name, storageType, expirationDate FROM
ItemSearchResult WHERE name = IFNULL(`$name`, name);
```

  - If user specifies filter of expiration date (`$expirationDate`)
    -

```
SELECT itemID, name, storageType, expirationDate FROM
ItemSearchResult WHERE expiration =
IFNULL(`$expirationDate`,expiration);
```

- If user specifies filter of storage type (`$storageType`)
  - ```
    SELECT itemID, name, storageType, expirationDate FROM
    ItemSearchResult WHERE storageType = IFNULL(`$storageType`,
    storageType);
    ```

- If user specifies filter of "Food"
  - ```
    SELECT
      Item.itemID, Item.name, Item.storageType, Item.expirationDate,
      Food.foodCategory
    FROM ItemSearchResult
    LEFT JOIN Food
    ON Item.itemID = Food.itemID;
    ```

- If user specifies filter of "Supply"
  - ```
    SELECT
      Item.itemID, Item.name, Item.storageType, Item.expirationDate,
      Supply.supplyCategory
    FROM ItemSearchResult
    LEFT JOIN Supply
    ON Item.itemID = Supply.itemID;
    ```

- If user specifies filter of individual category (`$individualCategory`)
  - If `$individualCategory` is one of food category

    ```
    SELECT
      Item.itemID, Item.name, Item.storageType, Item.expirationDate,
      Food.foodCategory
    FROM ItemSearchResult
    LEFT JOIN Food
    ON Item.itemID = Supply.itemID
    WHERE Food.foodCategory = $individualCategory;
    ```

  - If `$individualCategory` is one of supply category

    ```
    SELECT
      Item.itemID, Item.name, Item.storageType, Item.expirationDate,
    ```
    ```

```
  Supply.supplyCategory
FROM ItemSearchResult
LEFT JOIN Supply
ON Item.itemID = Supply.itemID
WHERE Supply.supplyCategory = $individualCategory;
```

- On each item from the item search result, user can click on ***Request Item*** button to specify number of items they want to request.
    - Look up user`s site table, if it is the same site as the Food bank, return message "cannot request from the same Food Bank as you are associated with."
    - Else, user can modify the number of that item to whatever they want to request.
        - If the number is zero, the item will be removed from Request Item table. (optional)

Retrieve Outstanding Requests

Abstract Code

- Click ***View Outstanding Requests*** button from **User Home** form.
    - If the user's associated Site does not have a food bank, return message "Unable to view Outstanding Request Form".
        - To get user's associated siteID

```
SELECT siteID FROM User WHERE username = $username;
```

        - To get food bank of that siteID

```
SELECT facilityName FROM Foodbank WHERE siteID = $siteID;
```

- Jump to **Retrieve Outstanding Requests** task to show **Outstanding Request Form**.
    - By default, result in **Outstanding Requests** Form is sorted by `$StorageType`.
        -

```
SELECT
    Request.requestID,
    Request.status,
    Request.quantityRequired,
    Request.quantityFulfilled,
    Item.itemName,
    Item.storageType,
    Item.itemType,
    Item.expirationDate
FROM Request LEFT JOIN Item
ON Request.itemID = Item.itemID
```

```
WHERE Request.facilityName = $facilityName
ORDER BY storageType;
```

- Click ***Sorted by Storage Type*** button to view the report Sorted by `$StorageType`.
    - SQL similar to the above SQL sorted by storage type.
- Click ***Sorted by Category button*** to view the report Sorted by `$Category`.
    - SQL similar to the above SQL sorted by storage type.
- Click ***Sorted by Subcategory*** button to view the report Sorted by `$SubCategory`.
    - SQL similar to the above SQL sorted by storage type.
- Click ***Sorted by Quantity of Items*** button to view the report Sorted by `$Quantity`.
    - SQL similar to the above SQL sorted by storage type.
- At each of the above subtask:
    - if more than one request for a particular item exists such that the total number of items requested is greater than the total number available, call **Mark Outstanding** Item subtask to mark those items in red.
        - Group By itemID to sum up all quantity required for an itemID. Then compare the sum with available quantity to see if it's outstanding.

```
SELECT
   Request.itemID,
   SUM(Request.quantityRequired) AS itemRequestSum,
   Item.availableQuantity
FROM Request LEFT JOIN Item
ON Request.itemID = Item.itemID
WHERE Request.facilityName = $facilityName
GROUP BY Request.itemID;
```

Add Request

Abstract Code

- On **Item Report** Form, once user sees the available items they are interested, user click on ***Add Request*** button to add an request
    -

```
INSERT INTO Request (requestID, itemID, status, quantityRequested,
quantityFulfilled) VALUES
(`$requestID`,`$itemID`,`$status`,`$quantityRequested`,`$quantityFulfill
ed`);
```

Update Request

Abstract Code

- On **Outstanding Requests** Form, user select the way to update the request.
  - If ***Mark Request Fulfilled in Full*** button is selected, look up that request in **Request** Form and update attribute Fulfilled Number (`$FulfilledNumber`) to whatever request number is made.
    -

```
UPDATE Request SET quantityFulfilled = `$quantityRequested`
WHERE requestID = `$requestID`;
```

  - If ***Reduce Request*** button is selected, look up that Request in **Requests** Form update `$FulfilledNumber` to the fulfillment number that user typed in.
    -

```
UPDATE Request SET quantityFulfilled = `$quantityFulfilled` WHERE
requestID = `$requestID`;
```

Retrieve Request

Abstract Code

- From **User** Form, Click ***Retrieve Requests Status*** button. Jump to **Retrieve Requests Status** task to show all requests this user has made.
  -

```
SELECT
  Request.userName,
  ClientService.facilityName,
  Request.itemName,
  Request.status,
  Request.quantityRequest,
  Request.quantityFulfilled
FROM Request
LEFT JOIN ClientService
ON Request.facilityID = ClientService.facilityID
WHERE Request.userName = $userName;
```

- Look up Request table and return status (pending or closed) attribute.
  - If status is closed, return actual number provided.

Cancel Request

Abstract Code

- Click ***Cancel Request*** button from **View Request Status** form. Jump to **Cancel Request** task.
- Look up the request that user is going to cancel from Request table.

- If the request is in the Outstanding Request Table, remove the request from Request table and Outstanding Request Table.
-

```
DELETE FROM Request
WHERE requestID = $requestID;
```