Table of Contents:

SQL Data Type:
Database Schema: *CS6400_sp17_team22.sql*
Table: User

User_ID int(16) unsigned NOT NULL AUTO_INCREMENT
UserName varchar (50) NOT NULL,
Password varchar (50) NOT NULL,
Email varchar (250) NOT NULL,
LastName varchar (100) NOT NULL,
FirstName varchar (100) NOT NULL

Table: Item
Item_ID int(16) unsigned NOT NULL AUTO_INCREMENT
Name varchar (50) NOT NULL,
Unit int(16) unsigned NOT NULL,
StorageType varchar (50) NOT NULL,
ExpirationDate DATE NOT NULL,
Category varchar (50) NOT NULL,
subCategory varchar (50) NOT NULL

Table: Site
Site_ID int(16) unsigned NOT NULL AUTO_INCREMENT
Name varchar (50) NOT NULL,
Phone_Number varchar (50) NOT NULL,
Street_Address varchar (250) NOT NULL,
City varchar (50) NOT NULL,
State varchar (50) NOT NULL,
Zipcode int(16) unsigned NOT NULL

Table: Client
Client_ID int(16) unsigned NOT NULL AUTO_INCREMENT
Name varchar (50) NOT NULL,
Phone_Number varchar (50) NOT NULL,
ID_Description varchar (250) NOT NULL,
Head_of_Household Boolean NOT NULL,
Field_Modified varchar (50) NOT NULL,
Service_Site_ID int(16) unsigned NOT NULL,
Service_Description varchar (50) NOT NULL,
Service_Date_Time timestamp NOT NULL

Table: Service
Service_Name varchar (50) NOT NULL

Table: SubService
Description varchar (250) NOT NULL,
Hours_Of_Operation Use varchar (100) NOT NULL,
Condition_For_Use varchar (250) NOT NULL

Table: Soup Kitchen
Seat_Capacity int(16) unsigned NOT NULL,
Seat_Available_Count int(16) unsigned NOT NULL

Table: Shelter
Familyroom_Count int(16) unsigned NOT NULL

Table: Bunk
Bunk_ID int(16) unsigned NOT NULL,
Type varchar (100) NOT NULL,
Bunk_Count int(16) unsigned NOT NULL,
Bunk_Available_Count int(16) unsigned NOT NULL

Table: Request

User_ID int(16) unsigned NOT NULL,
Item_ID int(16) unsigned NOT NULL,
Status varchar (100) NOT NULL,
Number_Requested int(16) unsigned NOT NULL,
Number_Prvided int(16) unsigned NOT NULL

Table: Waitlist
Site_ID int(16) unsigned NOT NULL,
Client_ID int(16) unsigned NOT NULL,
Waitlist_Ranking int(16) unsigned NOT NULL,
Date_Time DATETIME NOT NULL

Constraints:

User

- User must have a login account.
- User can modify data of the site only to which they are associated.
- Users must not be able to remove the last service associated with their site.
- Users at any site cannot modify the room waitlist at any other site.
- Users do not have full accessibility of the full list of all clients.
- The user cannot delete client, but user can modify the client's data.
- User should not be able to request an item from the food bank at their own site.

Site

- A site must provide at least one service.
- Reports other than the available room and meal remaining report, are only provided to logged in user only.

Client

- A search string must be provided to match at least one client in the database that may not match more than 5 clients in database.

Items

- Number of items requested while in pending status and items provided while in closed status must be tracked.
- Shortfall of an item in inventory should be highlighted in red or some other method.

Login

Task Decomp

Lock Types: Read-only on User table

Enabling Conditions: None
Frequency: Frequent

Schemas: Single

Consistency: not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User enters username ($Username), *password* ($Password) input fields.
- If data validation is successful for both *username* and *password* input fields, then:
    - When ***Enter*** button is clicked:
        - If User record is found but User.password != '$Password':
            - Go back to **Login** form, with error message.
        - Else:
            - Store login information as session variable $UserID.
            - Go to **Main Menu** form.
- Else username and *password* input fields are invalid, display **Login** form, with error message.

Outstanding Request Report

Task Decomp

Lock Types: View Report read-only, Edit Report can update the database

Enabling Conditions: Triggered by successful login
Frequency: View Report and Edit Report can have different frequencies

Schemas: multiple, request and items tables

Consistency: Important, reports should show the most updated numbers
Subtasks: Mother Task is needed.

Abstract Code

With a *User* logged in, validate whether the *User* is associated with a site that has a Food Bank. If yes, then:

If **View Report** button is pressed, then:

Display $item.name, $item.unit, $item.storagetype, $item.category and $item.subcategory.

Requests can be sorted by choosing the sorting column, storage type, category or sub-category, quantity of items.

Highlight numbers in red if there is shortfall in inventory for an item

If the *User* pressed the **Edit** button, then:

The *User* can mark the request fulfilled in full or edit the number of items

Items table is updated

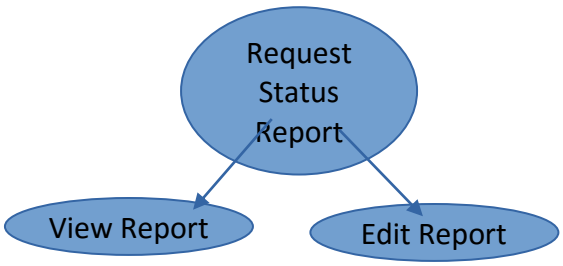
Else:

Do nothing

Else display error message

Request Status Report

Task Decomp



Lock Types: Read and edit on Request table

Enabling Conditions: Triggered by successful login
Frequency: view and update have different frequencies

Schemas: Single

Consistency: Important, most updated numbers are needed
Subtasks: Mother Task is needed

Abstract Code

With a *User* logged in, if he click on the **Request Status Report** button, then:

Validate if the *User* has made any request for items. If so, then:

Show the report with $item.name, $item.unit.

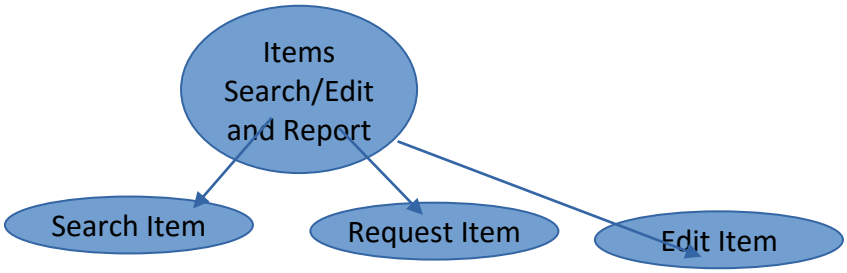
` If the *User* click on **Cancel** button, then:

Remove the request from Request table in the database

Else do nothing

Items Search/Edit and Report

Task Decomp



Lock Types: Read Update, Delete for Items table, Read, Insert for Request Table

Enabling Conditions: Triggered by successful login
Frequency: Frequent. View, Request and Edit will have different frequencies

Schemas: Multiple

Consistency: Important, most updated numbers are needed
Subtasks: Mother Task is needed

Abstract Code

With a *User* logged in, The *User* can enter a string to search for an item. If the *input* is valid, then:

Display all inventory ($item.name, $item.unit) for that Item. If the *input* is empty, display all inventory.

If the *User* choose filters, display the filtered inventory based all the filtering criteria

If the *User* click on the **Request** button, Request table is updated

Validate if the *User* is Associated with a Food Bank. If so, then:

The *User* can Edit the number of that Item.

If the count reaches zero, Items table is updated in the database

Else display error message

View Available Bunks/Rooms

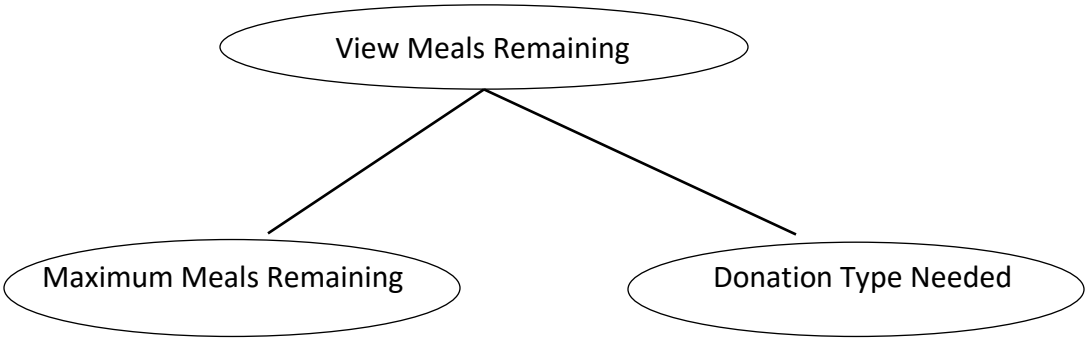
Task Decomposition:
Lock Types: Read-only on Site, SubService, Shelter, and Bunk.
Enabling Conditions: Enabled by clicking Available Room button on main page.
Frequency: Low – 100 per day
Schema: Multiple schemas needed.
Consistency: Critical, order is not critical.
Subtasks: Mother task is not needed. No decomposition needed.

Abstract Code

- Run the **View Available Bunks/Rooms** task.
- If bunks are available then
  Find the current Site, based on *$Site_ID*
  Display the current Site_Name, Physical_Location, PhoneNumber
  Find the current SubSevice
  Display Hours_of_operation, Condition_for_Use
  Find the current Bunk
  Display Available Count
  Else
  Display "Sorry all shelters are currently at maximum capacity".

View Meals Remaining

Task Decomposition:



Lock Types: Read-only, on Item table.
Enabling Conditions: Enabled by clicking Meals Remaining button on main page.
Frequency: Sub tasks have different frequency. Usually low.
Schema: One schema needed
Consistency: Critical, order is not critical
Subtasks: All tasks can be done in parallel. Mother task is required to coordinate subtasks.

Abstract Code

- Run the **View Meals Remaining** task.
- Maximum Meal Remaining:
  Fetch Name, Unit and Sub Category from Item table.
  If two items have same sub category

Add units of these two items.
Sort items in ascending order based on units
1st item's unit in the sorted items list will be the maximum remaining meals.
Display Maximum Meals Remaining.

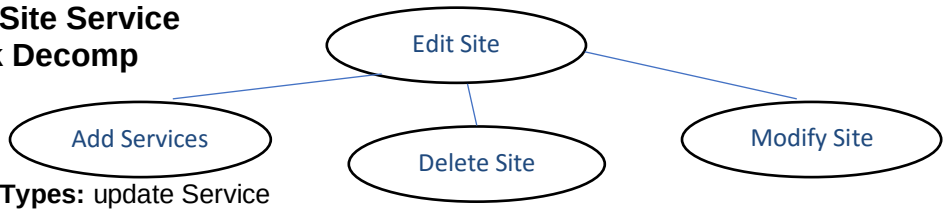
- Donation Type Needed:
  Fetch Name, Unit and Sub Category from Item table.
  If unit of item < required number of item's unit in the foodbank
  Fetch the sub category of item.
  Display Donation Type needed.

View Site Services Task Decomp

Lock Types: Read-Only
Enabling Condition: Triggered by user's successful login
Frequency: low
Schemas: multiple schema needed
Consistency: not critical
Subtask: Mother task is not needed. No decomposition needed

Abstract Code:
Run the ***View Site Services*** task based in $Site_ID
Find the current Site using Site_ID
Find each service for the site
{Display Service_Name;
If **subService** button is clicked:
Display Description, Hours of Operation, Condition for Use;
If Servie_Name is Soup Kitchen, display Seat_Capacity and Seat_Available.
If Servie_Name is Shelter, display FamilyRoom_Count, Bunk Type, Bunk_ID, Count and Available Count
}

Edit Site Service Task Decomp



Lock Types: update Service
Enabling Condition: Triggered by user's successful login
Frequency: low
Schemas: multiple schema needed
Consistency: Not critical
Subtask: Mother task is needed.

Abstract Code:
Run the **Edit Site Services** task based in $Site_ID
View Site Services, populate service dropdowns
If **ADD** button is pressed: **View Site Services**; display the form with room for another service;
If **EDIT** button is pressed: display the form with current services to be modified.
If **DELETE** button is pressed:

If only one service available, display error message 'Last service can only be deteted by database Administrator'
Else **Delete Service**; **View Site Services**;
If **SAVE** button is pressed: save change. **View Site Services.**

Search Clients

Task Decomposition:
Lock Types: Read-only on Client table
Enabling Conditions: Logged in as user
Frequency: Frequent
Schemas: One schema needed
Consistency: Not critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Display text box *input*($input)
Click on button ***search***:
Search and validate text box's content
$client_list= Find all clients with Name like *$input*
If count($client_list)>5, return nothing and pop out "warning: refine the search keyword"
Else if count($client_list) == 0, go to **Client Register Form**.
Else
For each $client
Find and display $Client.Client_ID, $Client.Name, $Client.ID_Description.
Click on $Client.Client_ID, will let you go to **Client Report**.

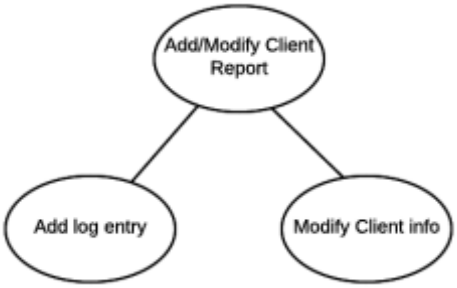
View Client Report

Task Decomposition:
Lock Types: Read-only on Client table
Enabling Conditions: Logged in as user
Frequency: Frequent
Schemas: One schema needed
Consistency: Not critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Run the **View Client Report** task based on $Client.Client_ID.
Find the current client's Name, Phone_Number, ID_Description, Head_of_Household and Log history.
Display the current client's Name, Phone_Number, ID_Description, Head_of_Household and Log history.

Add/Modify Client Report

Task Decomposition:



Lock Types: Write on Client table
Enabling Conditions: Logged in as user
Frequency: Frequent
Schemas: One schema needed
Consistency: Critical, order is not critical
Subtasks: Subtask can be done in parallel. Modify Client info button in main frame of client form, and add log entry at the end of client form.

Abstract Code:
Display **View Client Report** with text box (can be edited).
Display ***edit*** button next to each attribute (current client's Name, Phone_Number, ID_Description, Head_of_Household,).
Display text box and ***add log*** button at the end of client form.
When ***edit*** Button is pressed:
Validate input data type and length constraint.
Write and update the related attributes of current client.
When ***add log*** button is pressed:
Validate input data type and length constraint.
And a new log entry and write the attributes (*Field Modified, Site_ID, Date/Time, Discription*).

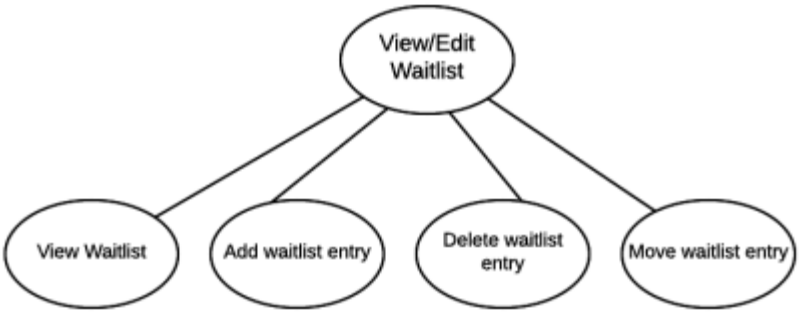
Add Client

Task Decomposition:
Lock Types: Write-only on Client table
Enabling Conditions: Logged in as user
Frequency: less Frequent
Schemas: One schema needed
Consistency: Critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Display the **Client register Form** with textbox of *Name, Phone_Number, ID_Description, Head_of_Household and Log(Field Modified, Site_ID, Date/Time,Description)* attributes.
When ***submit*** Button is pressed:
Validate input data type and length constraint.
Write and update the related attributes of current client.

View/Edit Waitlist

Task Decomposition:



Lock Types: Read/write on waitlist, read on shelter, read on client.
Enabling Conditions: Logged in as user with shelter
Frequency: Frequent
Schemas: Multiple
Consistency: Not important
Subtasks: 'View Waitlist' must be done first, and then the other tasks can be done in parallel. View waitlist, add waitlist and delete waitlist button in main frame of waitlist form. For each $waitlist entry, button ***delete***, ***move up*** and ***move down*** will provide.
And add waitlist at the end of client form.

Abstract Code:
View waitlist:
Determine waitlist associated with shelter and associated with client.
For each $waitlist on waitlist, sorted by $waitlist.Waitlist_Ranking.
Display $waitlist on waitlist, sorted by $waitlist.Waitlist_Ranking.

Delete waitlist entry:
If ***delete*** button is pressed:
  Delete the current entry in waitlist relationship table.
  For each $entry with position below $current_entry:
    Decrement $selected_entry.Waitlist_Ranking by 1.

Move waitlist entry:
If ***move up*** button is pressed:
  Validate input and check whether the current entry is at the top.
  Set $previous.Waitlist_Ranking to $current_ entry.Waitlist_Ranking.
  Set $current_entry. Waitlist_Ranking to $new_position
  Set $new_position to $previous. Waitlist_Ranking
If ***move down*** button is pressed:
  Validate input, check whether the current entry is at the bottom.
  Set $next.Waitlist_Ranking to $current_ entry.Waitlist_Ranking.
  Set $current_entry.Waitlist_Ranking to $new_position
  Set $new_position to $next.Waitlist_Ranking

Add waitlist entry:

If ***add*** button is pressed:

Validate input data type(date/time) and length constraint.

Write a new entry with attributes of *client_id, service_id* and *date/time* into waitlist relationship table.