Phase 1 Report

CS 6400 - Fall 2016 Team 010

Atlanta Sharing Alliance Coordination System (ASACS) Data Types

SQL Data Types

Database Schema: cs6400_sp17_team010.sql

Table: Client

‘Client ID Number and Description’	VARCHAR(300) NOT NULL UNIQUE
‘Client ID Number’	VARCHAR(30) NOT NULL
‘Client ID Description’	VARCHAR(255) NOT NULL
‘Full Name’	VARCHAR(60) NOT NULL
‘First Name’	VARCHAR(30) NOT NULL
‘Last Name’	VARCHAR(30) NOT NULL
‘Phone Number’	VARCHAR(12)

Table: User

‘Username’	VARCHAR(30) NOT NULL UNIQUE
‘Password’	VARCHAR(30) NOT NULL
‘Full Name’	VARCHAR(60) NOT NULL
‘First Name’	VARCHAR(30) NOT NULL
‘Last Name’	VARCHAR(30) NOT NULL
‘Email’	VARCHAR(30) NOT NULL UNIQUE

Table: Log

‘Date Time’	TIMESTAMP DEFAULT CURRENT_TIMESTAMP
‘Site’	SMALLINT UNSIGNED NOT NULL
‘Service Description’	VARCHAR(500) NOT NULL
‘Notes’	VARCHAR(500)

Table: Site

‘Name and Location’	VARCHAR(200) NOT NULL UNIQUE
‘Name’	VARCHAR(100) NOT NULL
‘Location’	VARCHAR(100) NOT NULL
‘Street Address’	VARCHAR(100) NOT NULL
‘City’	VARCHAR(50) NOT NULL
‘State’	VARCHAR(2) NOT NULL
‘Zip Code’	VARCHAR(10) NOT NULL
‘Primary Phone’	VARCHAR(12) NOT NULL UNIQUE

Table: Item

‘Name/Expiration Date’	VARCHAR(100) NOT NULL UNIQUE
‘Name’	VARCHAR(30) NOT NULL
‘Number of Unit’	SMALLINT UNSIGNED NOT NULL
‘Unit Type’	ENUM(‘Bag’, ‘Box’, ‘Carton’, ‘Others’) NOT NULL
‘Expiration Date’	TIMESTAMP
‘Storage Type’	ENUM(‘Dry Good’, ‘Refrigerated’, ‘Frozen’) NOT NULL

Table: Supplies

‘Type of Supplies’	ENUM(‘Personal hygiene’, ‘Clothing’, ‘Shelter’, ‘Other’) NOT NULL

Table: Food

‘Type of Food’	ENUM(‘Vegetables’, ‘Nuts/Grains/Beans’, ‘Meat/Seafood’, ‘Dairy/Eggs’, ‘Sauce/Condiment/Seasoning’, ‘Juice/Drink’) NOT NULL

Table: Request

‘Request Date Time’	TIMESTAMP DEFAULT CURRENT_TIMESTAMP
‘Count of Request’	SMALLINT UNSIGNED NOT NULL
‘Count of Provided’	SMALLINT UNSIGNED NOT NULL
‘Status’	ENUM(‘Filled’, ‘Partially Filled’, ‘Unable to be Filled’, ‘Pending’) NOT NULL DEFAULT ‘Pending’

Table: Services

‘Description’	VARCHAR(300) NOT NULL UNIQUE
‘Name’	VARCHAR(100) NOT NULL
‘Address’	VARCHAR(200) NOT NULL
‘Hours Of Operation’	VARCHAR(255) NOT NULL
‘Conditions For Use’	VARCHAR(255) NOT NULL

Table: Waitlist

‘Waiting List Position’	TINYINT UNSIGNED NOT NULL

Table: Shelter

‘Rooms Available’	SMALLINT UNSIGNED NOT NULL

Table: Soup Kitchen

‘Seats Available’	SMALLINT UNSIGNED NOT NULL
‘Seat Capacity’	SMALLINT UNSIGNED NOT NULL

Table: Room

‘Room Number’	SMALLINT UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT

Table: Family Room

‘Occupation Status’	BOOLEAN

Table: Bunk Room

‘Bunks Available’	SMALLINT UNSIGNED NOT NULL
‘Bunks Capacity’	SMALLINT UNSIGNED NOT NULL
‘Bunk Types’	ENUM(‘Male’, ‘Female’, ‘Mix’) NOT NULL

Atlanta Sharing Alliance Coordination System (ASACS) Constraints

Constraints

Client

- New clients needs to be registered into system by user
- Each client can only be in system once
- Client cannot be deleted by user from system

User

- New user needs to be added in by DBA
- Each user can only be in system once

Log

- Must capture which services are used at which sites and by which client, along with timestamp
- Must capture client name changes
- Must capture changes to description field

Site

- Sites can only be added/deleted by DBA
- A site cannot have more than one of each type of service
- Only user associated with this site can add or modify services of this site
- The last or only service of a site cannot be removed by the associated user

Item

- Items are added into system to a food bank by user associated with the food bank
- An item must have a name, number of units, and expiration date (non-expiring items are set to 01/01/9999)

Supplies

- A supplies item must have a storage type of one of the following: dry good, refrigerated, or frozen
- A supplies item must have a category type of one of the following: personal hygiene, clothing, shelter, or other

Food

- Food item must be set to one of the following category types: vegetables, nuts/grains/beans, meat/seafood, dairy/eggs, sauce/condiment/seasoning, or juice/drink.
- A “meal” is one unit each of a vegetable, nuts/grains/beans, and meat/seafood or dairy/eggs.

Request
- Requested item/food must exist in table Item
- Requester must be a user of the system associated with a food pantry, shelter, or soup kitchen, and not associated with the food bank that stores the item
- Request has to be made to an existing food bank
- Detail of request must be recorded: item name, quantity requested, food bank chosen, user/requester

Service
- A service type cannot exist at a site more than once

Waitlist
- Waitlist for a given Shelter can only be seen by a User associated with that Shelter’s Site.

Shelter
- Waitlist must be used to manage family rooms occupation

Soup Kitchen
- Soup kitchen has maximum seating capacity and tracks seating availability
- Seating availability cannot be higher than seating capacity

Food Pantry
- A food pantry must be a reference to a service

Food Bank
- Client cannot interact directly to a food bank
- Requested quantity of an item/food cannot exceed availability

Reports
- Web reports (Bunks/Rooms Report and Meals Report) are accessible to anyone (no authentication required)
- All other reports require authenticated user login

Task Decomposition with Abstract Code

Authenticate

Task Decomposition

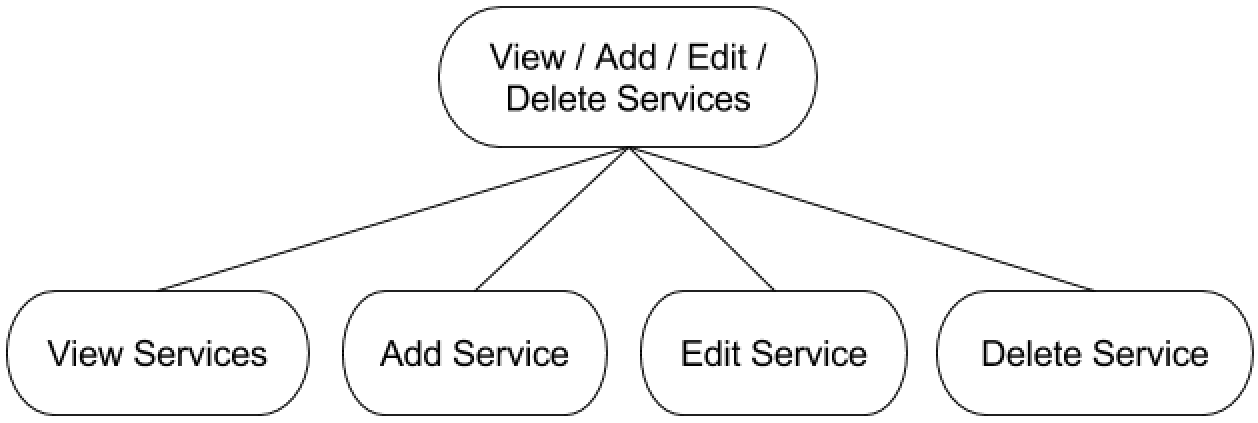
Lock types: Read only on User table
Enabling condition: None -- login form is publicly accessible entry point
Frequency: About 100 / day
Schemas: Single
Consistency: Not critical
Subtasks: Mother task is not needed. No decomposition required.

Abstract Code

- User enters *username*, *password* input fields.
- If *username* and *password* fields are both valid, then:
    - When ***Enter*** button is clicked:
        - Store login information as $Username variable
        - Go to ***Available Services Report*** form
- Else:
    - Go back to ***Login Form*** with error message

View / Add / Edit / Delete Services

Task Decomposition



Lock types: Read-only for View, write for Add / Edit / Delete on Site and Services tables
Enabling condition: All enabled by User login, Add / Edit / Delete enabled by separate requests
Frequency: View very often, Add / Edit / Delete rarely
Schemas: At least two -- one Site, at least one Service
Consistency: Not critical
Subtasks: View first, then Add / Edit / Delete may be done in any order. Mother task is needed.

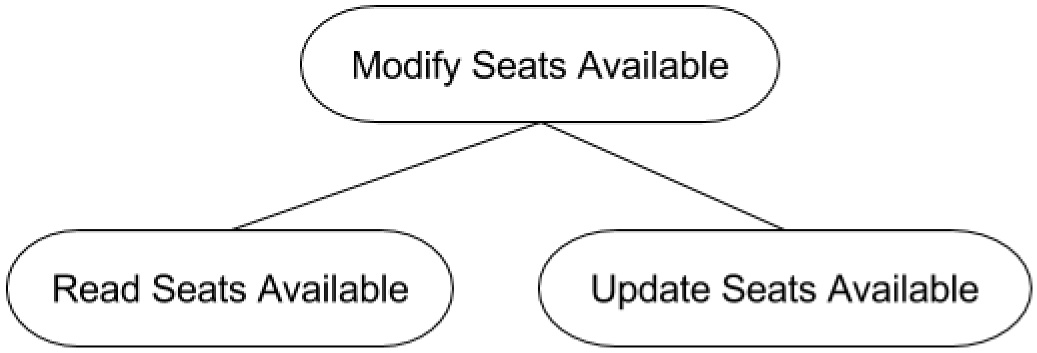
Abstract Code

- 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
    - 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
            - 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)
                - 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:
        - 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:
        - Go to ***Bunk Count Report*** form.
- Press ***Update This Service*** button:
    - **Edit Service** task using *description, hours, conditions* input fields
    - **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
            - **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
        - **View Services** task

Modify Seats Available

Task Decomposition



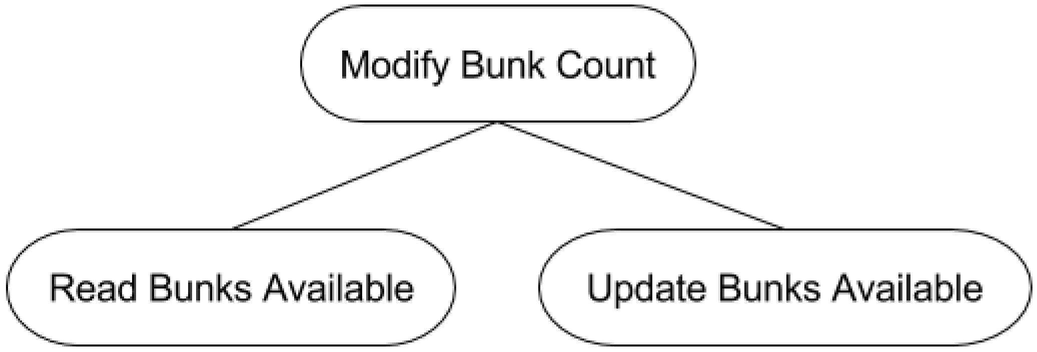
Lock types: Read / update on Soup Kitchen table
Enabling condition: Button press from Available Services Report; passes $Service; authenticated User logged in
Frequency: About 20 times / day for both
Schemas: Single
Consistency: Not critical
Subtasks: Read current value first, then modify. Mother task is needed.

Abstract Code

- Find current Soup Kitchen
- **Read Seats Available** task.
- Display seat capacity.
- Populate *seats available* input field with current value.
- Show ***Update*** button.
- On ***Update*** button press:
    - If *seats available* field is valid (less than capacity):
        - **Update Seats Available** task**.**
        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message

Modify Bunk Count

Task Decomposition



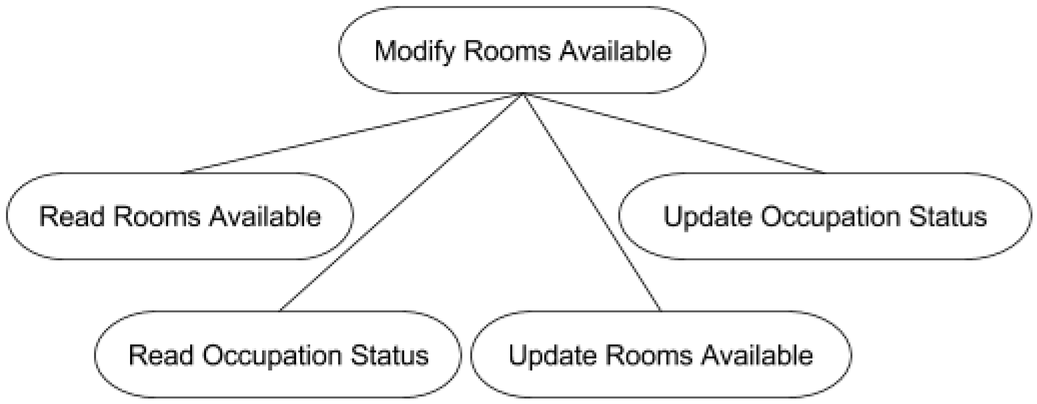
Lock types: Read from Shelter table and Bunk Room table; write to Bunk Room table
Enabling condition: Button press from Available Services Report; passes $Service; authenticated User logged in
Frequency: About 20 times / day for both
Schemas: At least two -- Shelter, at least one Bunk Room
Consistency: Not critical
Subtasks: Read current value first, then modify. Mother task is needed.

Abstract Code

- Find current Shelter.
- Find all Bunk Rooms associated with the Shelter.
- For each Bunk Room:
    - **Read Bunks Available** task.
    - Display bunk capacity.
    - Populate *bunks available* input field with current value.
- Show ***Update*** button.
- On ***Update*** button press:
    - If input value for *bunks available* field is valid (less than capacity) for all Bunk Rooms:
        - For each Bunk Room:
            - **Update Bunks Available** task**.**
        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message.

Modify Rooms Available

Task Decomposition



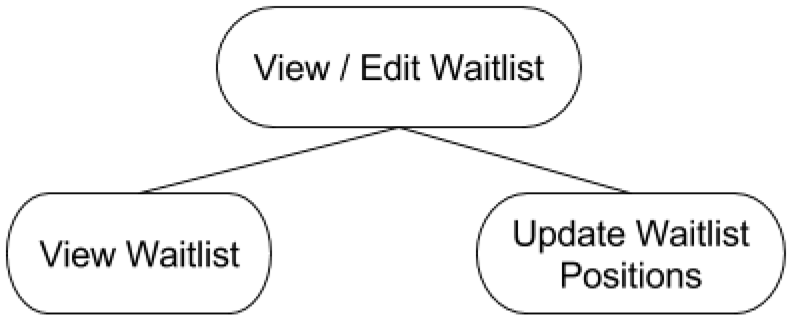
Lock types: Read from Shelter table and Family Room tables; update on both
Enabling condition: Button press from Available Services Report; passes $Service; authenticated User logged in
Frequency: Rare; same for all tasks.
Schemas: At least two -- Shelter, at least one Family Room
Consistency: Not critical
Subtasks: Both reads first, then modify. Table access order not important. Mother task is needed.

Abstract Code

- Find current Shelter.
- **Read Rooms Available** task.
- Populate *rooms available* input field with current value from Shelter.
- Find all Family Rooms associated with the Shelter.
- For each Family Room:
    - Populate *occupation status* input field with current value from Family Room.
- Show ***Update*** button.
- On ***Update*** button press:
    - If input value valid for all *rooms available* and *occupation status* fields:
        - **Update Rooms Available** task.
        - For each Family Room:
            - **Update Occupation Status** task**.**
        - Go to ***Available Services Report*** form.
    - Else:
        - Display error message.

View / Edit Waitlist

Task Decomposition



Lock types: Read on Shelter table to retrieve Clients, read on Clients for description information, and read on Waitlist table for Waiting List Position for View; write to Waitlist for Update
Enabling condition: Button press from Available Services Report; passes $Service; authenticated User logged in
Frequency: Medium for View, low for Edit
Schemas: Two -- Client and Shelter
Consistency: Not critical for View. Update must happen for all affected Clients at once (eg. should never be two Clients with same waitlist position when one Client is moved up in the waitlist).
Subtasks: View first, then Edit. Mother task is needed.

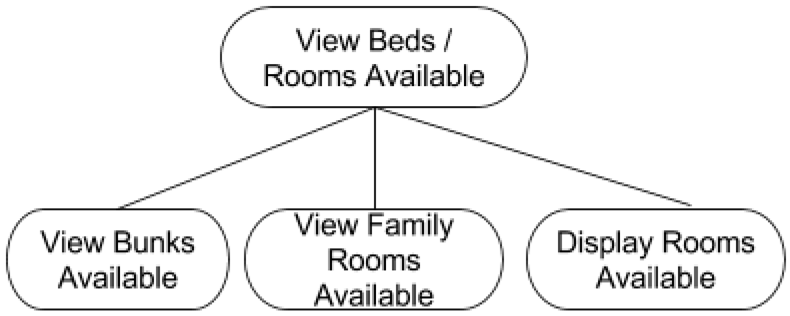
Abstract Code

- Find current Shelter.
- **View Waitlist** task: Find all Clients in Waitlist relationship with Shelter, sorted by Waitlist.Waiting List Position
- If no Waitlist relationships found:
    - Display "Waitlist is empty"
- Else:
    - For each Client:
        - Display Client.Full Name, Waitlist.Waiting List Position
        - Show *selection* radio button
    - Show ***Add, Delete, Move Up,*** and ***Move Down*** buttons
    - On ***Add*** button press:
        - Go to ***<u>Client Search Form</u>***
    - If *selection* is set:
        - On ***Delete*** button:
            - Ask for confirmation; if received:
                - Remove Client from Waitlist relationship

            - Move all other Clients below deleted Client up by one 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
            - Decrement Client’s waitlist position
        - 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
            - Increment Client’s waitlist position
        - Else: Display error message

View Beds / Rooms Available

Task Decomposition



Lock types: Read on Family Room table to determine available family rooms. Read on bunk rooms to determine available beds
Enabling condition: None; report is publicly available.
Frequency: Frequent, same for all tasks.
Schemas: One, basically just the Shelter schema
Consistency: Not critical for views, but Displaying rooms available must be done at same time as view otherwise, could display wrong information
Subtasks: Need mother task

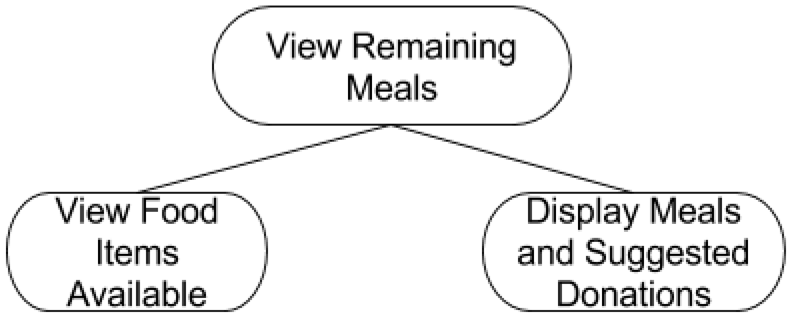
Abstract Code

- Run **View Beds / Rooms Available** task.
- Find family rooms that are unoccupied using the Occupation Status attribute
- Find Bunks available using the Bunks Available Status

- If no rooms or bunks are available:
    - Display “Sorry, all shelters are currently at maximum capacity”
- Else:
    - For each shelter with available room and/or bed:
        - display name, site location, phone number, hours of operation and conditions, number of male/female/mixed bunks and rooms available.

View Remaining Meals

Task Decomposition


View Remaining Meals
View Food Items Available
Display Meals and Suggested Donations

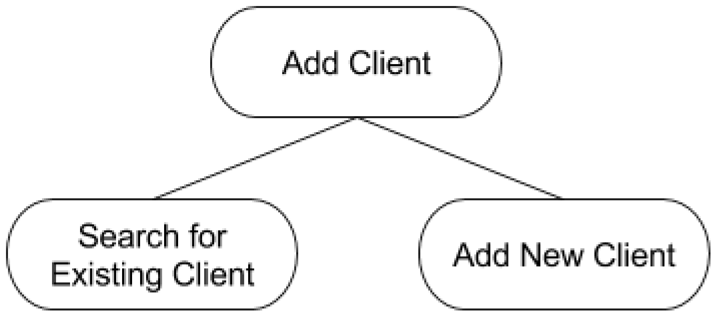
Lock types: Read on Items Table
Enabling condition: None; report is publicly available.
Frequency: Frequent; same for all tasks
Schemas: One, just the items schema.
Consistency: Not critical for view, but displaying meals and suggested donations must be done at same time as view or else wrong information displayed
Subtasks: Mother task is needed

Abstract Code

- Run **View Remaining Meals** task.
- Find Food items available that have not expired yet using number of unit and expiration date attributes
- Aggregate items into 3 categories:
    - Vegetables
    - Nuts/Grains/Beans
    - Meat seafood or Dairy/Eggs
- Calculate maximum number of meals as the minimum of the count of items in the three categories.
- Display minimum of the category as the type of donations most needed

Add Client

Task Decomposition



Lock types: Read on Client table for client search, add to Client table for add new.
Enabling condition: Authenticated User logged in; button press from Available Services Report.
Frequency: Search more often than add.
Schemas: Single - Client only.
Consistency: Not critical.
Subtasks: Search must happen first. Add may or may not follow, depending on search results. Mother task is needed.

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, last name, id number,* and *id description* fields for valid, non-null entries. If valid:
    - **Search for Existing Client** task: Find any Client with Client.First Name, Client.Last Name, Client.Client ID Number, and Client.Client ID Description that match the input fields.
    - If existing client found:
      - Display error message
    - Else:
      - **Add New Client** task: Add new entry to Client table, populating Client.First Name, Client.Last Name, Client.Client ID Number, Client.Client ID Description, and Client.Phone Number from the input fields
  - Else if not valid:
    - Display error message

Client Search

Task Decomposition

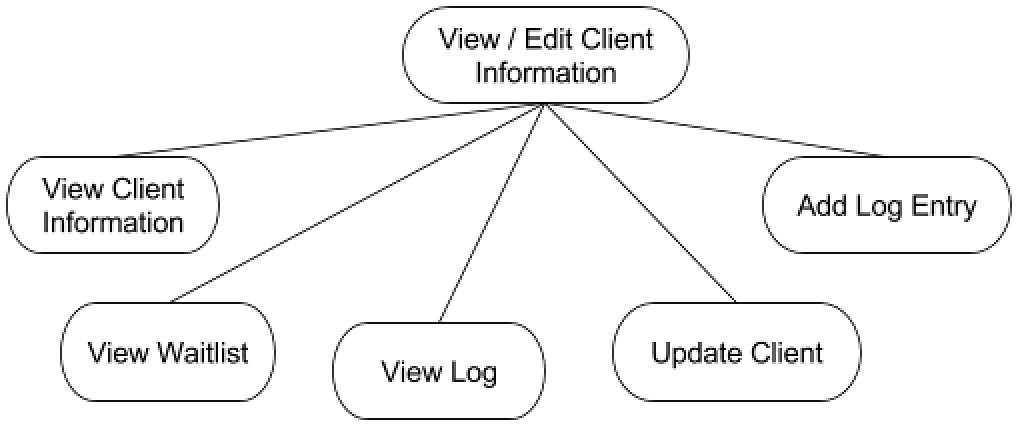
Lock types: Read on Client table for client description.
Enabling condition: Authenticated User logged in; button press from Available Services Report or the Waitlist Report.
Frequency: Very frequent.
Schemas: Single - Client only.
Consistency: Not critical.
Subtasks: No decomposition needed.

Abstract Code

- Display *client name/ID* input field.
- Run **Client Search** task: find all Clients with either Name or ID Number matching the input string.
- 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
        - Show ***Select*** button
- Else:
    - Display “No matching results” message.
- On ***Select*** button press:
    - Go to ***<u>Client Report</u>***, pass selected client as $Client

View / Edit Client Information

Task Decomposition



Lock types: Read-only on Client table for View, modify on Client table for Update, insert on Log table for log entry. Read-only on Waitlist and Shelter for View Waitlist. Read-only on Log for View Log.
Enabling condition: Authenticated User login; selection from Client Search Form passes $Client
Frequency: Medium for View, Low for Update/Log
Schemas: Client, Log, Waitlist, Shelter
Consistency: Not critical. Client's profile can be viewed while edits are made.
Subtasks: All three Views must be done first, but order is not critical. If update is, necessary, log entry must run first, then update, in order to store old values. Mother task is needed.

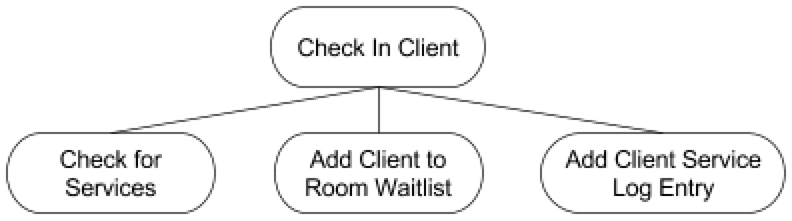
Abstract Code

- Find current Client from $Client.
- **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
  - **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
  - **View Log** task: Find all Log entries associated with the Client
  - For each Log entry:
    - Display Log.Date Time, Log.Site, Log.Service Description, Log.Notes
- 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 for Log.Date Time, current Site.Name for Log.Site, and “field modified” for Log.Service Description
        - **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

Task Decomposition



Lock types: Check for Services will need read on Site to determine available services. For adding client to waitlist, will need write to Client and Shelter tables. For adding a log entry, will need write to Log.
Enabling condition: For all: authenticated User login, Client selected via Client Search Form, which passes $Client. Add to Waitlist only enabled if current Site provides a Shelter.
Frequency: Check for Services often, Add Log often, Add to Waitlist rarely
Schemas: Client, Site, Shelter, Log
Consistency: Not critical, except that add to waitlist must check waitlist and add to it in one step.
Subtasks: Check for Services has to happen first. Add to Waitlist and Add Log can happen in any order. Mother task is needed.

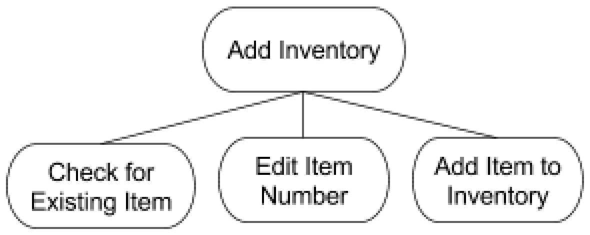
Abstract Code

- Find current Client from $Client
- Find current Site from $Site (set on login)
- **Check for Services** task with current Site
- For each available service:
    - Display Service.Description
    - 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
    - Else:
        - Display error message
- On ***Modify Bunk Count*** button:
    - Go to ***<u>Bunk Count Report</u>***

Add Inventory

Task Decomposition



Lock types: Read from Food and Supplies to retrieve current inventory. Modify to Food and Supplies if Item already exists, insert if it does not.
Enabling condition: Button press from Available Services Report; passes $ServiceID; authenticated User logged in
Frequency: Rare -- about once a day. Same for both tasks.
Schemas: Two -- Food, Supplies.
Consistency: Not critical; if identical item appears in inventory between Check and Add, there will just be two listings instead of one.
Subtasks: Check must be done first; then, either Edit or Add but not both. Mother task is needed.

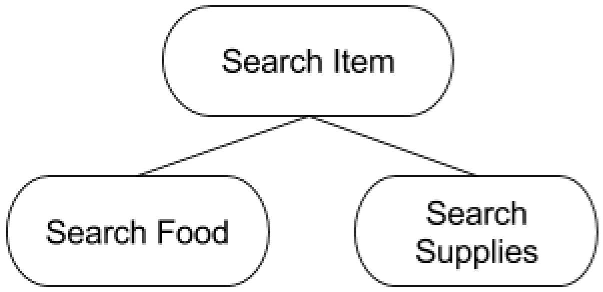
Abstract Code

- Create the following input fields:
    - *name*
    - *number of units*, drop-down list specifying *unit type* (Bag, Box, Carton, Others)
    - *expiration date*, populated with 01/01/9999 for default
    - *storage type* drop-down list (Dry Good, Refrigerated, Frozen)

    - *category* drop-down list (Food, Supplies)
    - *food type* drop-down list (Vegetables, Nuts/Grains/Beans, Meat/Seafood, Dairy/Eggs, Sauce/Condiment/Seasoning, Juice/Drink)
    - *supplies type* drop-down list (Personal Hygiene, Clothing, Shelter, Other)
- Show ***Add Inventory*** button
- On ***Add Inventory*** button press:
    - If all fields valid:
        - Get item type from *supplies type* field if category is Supplies, else get type from *food type* field
        - **Check for Existing Item** task: find the first current item associated with current Food Bank (from $Service) that matches name, unit type, expiration date, storage type, category, and item type entered
        - If existing item found:
            - **Edit Existing Item** task: Add *number of units* to existing number
        - Else:
            - **Add Item to Inventory** task: add new Item to either Food or Supplies table

Search Item

Task Decomposition



Lock types: Read on Item/Food and Item/Supplies
Enabling condition: Button press from Available Services report, passes $Service. Authenticated User of any Site logged in is required.
Frequency: High frequency, same for both.
Schemas: Two -- Item/Food, Item/Supplies.
Consistency: Not important.
Subtasks: Search can be done in either order, but may both have to be done. Mother task is needed.

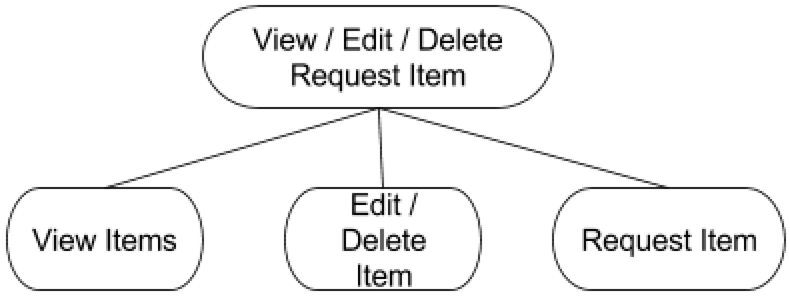
Abstract Code

- Retrieve Food Bank.Name from all Food Banks in the system
- Display input fields for:
    - *food bank* drop-down list, populated with the retrieved Food Bank names

        - *category* drop-down list (All, Food, Supplies)
        - *storage type* drop-down list (All, Dry Good, Refrigerated, Frozen)
        - *food type* drop-down list (All, Vegetables, Nuts/Grains/Beans, Meat/Seafood, Dairy/Eggs, Sauce/Condiment/Seasoning, Juice/Drink)
        - *supplies type* drop-down list (All, Personal Hygiene, Clothing, Shelter, Other)
        - *expiration date after* calendar selection widget
        - *name*
- Display ***Search*** button.
- On ***Search*** button press:
    - If *category* field contains 'Food' or 'All':
        - **Search Food** task: find all Foods matching input fields
    - If *category* field contains 'Supplies' or 'All':
        - **Search Supplies** task: find all Supplies matching input fields
- 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

Task Decomposition



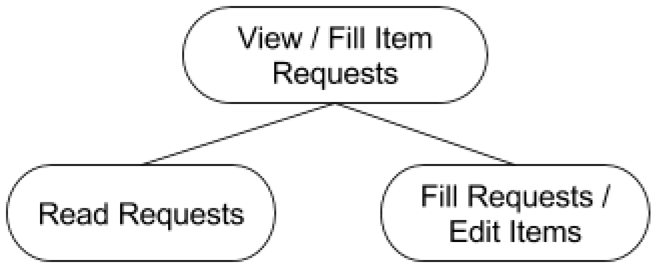
Lock types: Read on Item/Food and Item/Supplies for View. Insertion on Request for Request Item, update or delete to Item/Food or Item/Supplies for Edit Item.
Enabling condition: Search results ($Item) passed from ***Item Search Form***. Authenticated User of any Site logged in is required for all tasks. Edit Item requires a User associated with the Food Bank the Item is stored in. Request Item requires a User *not* associated with the Food Bank the Item is stored in, that *is* associated with a Food Pantry, Shelter, or Soup Kitchen.
Frequency: View high frequency, Request Item medium frequency, Edit Item low frequency.
Schemas: Three -- Item/Food, Item/Supplies, and Request.
Consistency: Not critical.
Subtasks: View must happen before Edit/Request. Edit and Request can be done in either order. Mother task is needed.

Abstract Code

- Find current User’s Site from $Site set on login
- Find any Food Bank services associated with current User’s Site, store as $Service
- **View Items** task: For each item in $Item:
    - Find current Item
    - Find Food Bank associated with Item
    - Display descriptive fields from Item.Name, Item.Category, Item.Expiration Date, Item.Storage Type, Food Bank.Name
    - If item is a Food:
        - Display Food.Type of Food
    - Else:
        - Display Supplies.Type of Supplies
    - If Food Bank is the same as $Service:
        - Display editable *number of item* input field, populated from Item.Number of Units
        - Display non-editable *request unit number* input field, populated with zero
    - Else if current Site does not have Food Pantry, Shelter, or Soup Kitchen:
        - Display non-editable Item.Number of Units
        - Display non-editable *request unit number* input field, populated with zero
    - Else:
        - Display non-editable Item.Number of Units
        - Display editable *request unit number* input field, initially populated with zero
- Display ***Update*** button
- On ***Update*** button press:
    - Find each Item for which Food Bank is the same as $Service ($EditItem)
    - Find each Item for which *request unit number* input field is non-zero ($RequestItem)
    - Display confirmation message listing each $EditItem, with Name and new Number, and each $RequestItem, with Name and requested number.
    - If confirmation received:
        - Run **Edit Item** task to update each $EditItem.Number of Unit from *number of item* input field. If new number is zero, the item should be deleted from the table.
        - Run **Request Item** to make new Request association between each $RequestItem and current User
- Re-run **View Items** task to refresh table

View / Fill Item Request

Task Decomposition



Lock types: Read-only on Request and Item for Read Requests. Update on Request for Fill Request. Update or delete on Item for Edit Item to update the number available.
Enabling condition: Authenticated Food Bank User logged in; click from Available Services Report.
Frequency: Read Requests more frequent. Fill Requests and Edit Items less frequent.
Schemas: Two - Request, Item.
Consistency: Fill Requests and Edit Items must be done together to make sure Inventory reflects granted requests, so can't be decomposed.
Subtasks: Read first, then Fill and Edit. Mother task is needed.

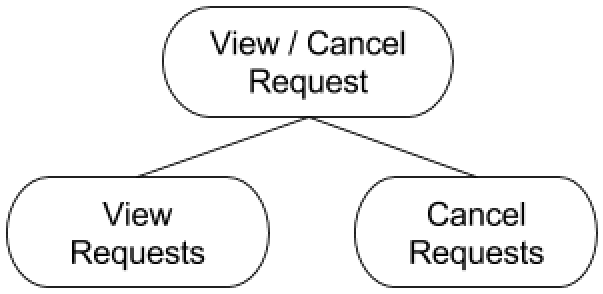
Abstract Code

- Find Food Bank associated with current user, from $SiteID set on login.
- **Read Requests** task: find all Requests associated with the current Food Bank.
- Display sortable columns: Name, Storage Type, Category, Sub-Category, Number Available, and Number Requested
- For each request:
  - Find the Item associated with the request.
  - Display the Item.Name, Item.Storage Type, category (Food or Supplies), Item.Type of Supplies (if Supplies) or Item.Type of Food (if Food), Item.Number of Unit, and the Request.Count of Request.
  - Show a *number provided* input field and a *status* drop-down list (Pending, Filled, Partially Filled, Unable to Fill). Populate *status* with Request.Status.
  - Find any other existing Requests associated with the same Item.
  - If the sum of these Request.Count of Request is higher than the Item.Number of Unit:
    - Highlight current request.Count of Request 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 Request.Status and set Request.Count of Provided from *status* and *number provided* fields
            - Find the Item associated with the request. Subtract the number provided from the Item.Number of Unit. If the result is zero, delete the Item.
        - **Read Requests** task to re-display Request list
    - Else:
        - Display error message

View / Cancel Item Request

Task Decomposition



Lock types: Read on Request and Item tables for View. Delete on Request table for Cancel.
Enabling condition: Authenticated User logged in; click from Available Services Report.
Frequency: View often, Cancel rarely.
Schemas: Three - Request, Food, and Supplies.
Consistency: Not critical.
Subtasks: View must happen before Cancel. Cancel may or may not be called. Mother task is needed.

Abstract Code

- Find current User from $Username, set on login.
- Find all Requests associated with current User.
- **View Requests** task:
- For each request:
    - Find the Item associated with the request.
    - Display the Item.Name, Item.Storage Type, category (Food or Supplies), Item.Type of Supplies (if Supplies) or Item.Type of Food (if Food), Item.Number of Unit, Request.Count of Request, Request.Count of Provided, and Request.Status.
    - Show a *cancel* check box.
- Show a ***Cancel*** button.

- On ***Cancel*** button press:
    - **Cancel Requests** task:
    - Find each Request with a *cancel* check box selected.
    - Ask user for confirmation. If received: delete the Request from the table.
    - **View Requests** to refresh table.