Table of Contents

Data Types:

User

Attribute	**DataType**	**Nullable**
Username	String	Not Null
Password	String	Not Null
FirstName	String	Not Null
LastName	String	Not Null
Email	String	Not Null
UserType	String	Not Null

Client

Attribute	**DataType**	**Nullable**
FirstName	String	Not Null
LastName	String	Not Null
Description	String	Not Null
Phone	String	Null

Site

Attribute	**DataType**	**Nullable**
SiteId	Number	Not Null
ShortName	String	Not Null
StreetAddress	String	Not Null
City	String	Not Null
State	String	Not Null
ZipCode	Number	Not Null
Phone	String	Not Null

ServiceList	List<String>	Not Null

Shelter

Attribute	**DataType**	**Nullable**
FacilityName	String	Not Null
HoursOfOperation	String	Not Null
Conditions for use	String	Not Null
BunkType	String	Not Null
BunkCountMale	Number	Not Null
BunkCountFemale	Number	Not Null
BunkCountMixed	Number	Not Null
EligibilityCondition	List<String>	Not Null

SoupKitchen

Attribute	**DataType**	**Nullable**
FacilityName	String	Not Null
HoursOfOperation	String	Not Null
SeatCount	Number	Not Null
EligibilityCondition	List<String>	Not Null

FoodPantry

Attribute	**DataType**	**Nullable**
FacilityName	String	Not Null
HoursOfOperation	String	Not Null
EligibilityCondition	List<String>	Not Null

FoodBank

Attribute	**DataType**	**Nullable**
FacilityName	String	Not Null

Item

Attribute	**DataType**	**Nullable**
ItemId	String	Not Null
Name	String	Not Null
ExpirationDate	Date	Not Null
ItemType	String	Not Null
StorageType	String	Null

Food

Attribute	**DataType**	**Nullable**
ItemId	String	Not Null
FoodCategory	String	Not Null

Supply

Attribute	**DataType**	**Nullable**
ItemId	String	Not Null
SupplyCategory	String	Not Null

Request

Attribute	**DataType**	**Nullable**
RequestId	Number	Not Null
ItemId	String	Not Null
Status	String	Not Null
QuantityRequested	Number	Not Null

QuantityFulfilled	Number	Null

ItemFoodbank

Attribute	**DataType**	**Nullable**
FacilityName	String	Not Null
ItemId	String	Not Null
AvailableQuantity	Number	Not Null

UserSite

Attribute	**DataType**	**Nullable**
Username	String	Not Null
SiteId	Number	Not Null

ClientService

Attribute	**DataType**	**Nullable**
ClientId	String	Not Null
SiteId	Number	Not Null
UserId	String	Not Null
DateTime	String	Not Null
Description	String	Not Null
Note	String	Null

ClientLog

Attribute	**DataType**	**Nullable**
ClientId	String	Not Null
UserId	String	Not Null
DateTime	String	Not Null

FieldModified	String	Not Null
PreviousValue	String	Not Null

Business Logic Constraints:

- Users at a site with a food bank can approve or edit requests for their site's food bank.
- Users of a site are free to add additional services, or modify existing services. System must allow them to create, delete and edit any of services for their site (only).
- Users should not be able to remove the last service associated with their site. They will need to request that the database administrator removes the entire site for them.
- When clients check into shelters a user of the ASACS will use the system to review their usage log (making sure they qualify based upon that shelters' conditions) and reduce the available bunk count as appropriate.
- All users of the ASACS system who are associated with a Shelter, Food Pantry or Soup Kitchen can put in a request for one or more (up to the maximum amount available) item from a food bank.
- A request for an item will have a source site / Food Bank, as well as a destination User (who made the request).
    - It can be in "pending" status (if it has not been acted upon by a user from the source food bank)
    - It can be in "closed" status if the request was fully or partially fulfilled.
    - Request must keep track of the number of items requested while in pending status, and the number of items provided once it is in closed status.
    - It is possible for an item to be removed from the ASACS database (because its stock quantity reached zero) while a pending request for that item still exists. In this case, the request should be marked as closed with zero items provided.
- Meal is defined as one unit each of a Vegetable, nuts/grains/beans, and Meat/seafood or Dairy/eggs.
- Food types must be from these categories:
    - Vegetables
    - Nuts/grains/beans
    - Meat/seafood
    - Dairy/eggs
    - Sauce/Condiment/Seasoning
    - Juice/Drink
- Storage type must be from these categories:

  - Dry Goods
  - Refrigerated
  - Frozen

- Supply type must be from these categories:
  - Personal hygiene
  - Clothing
  - Shelter
  - Other

- A site must provide at least one service.

- A site can not provide more than one service of a particular service type.

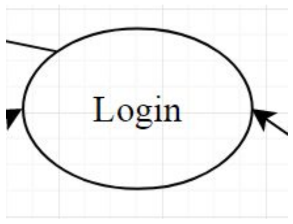
- A site may have no more than 4 services.

- Available Bunks and Meals Remaining reports must have no user authentication.

Login

Task Decomposition

Lock Types: Read-only on User table
Number of Locks: Single
Enabling Conditions: None
Frequency: Frequent
Consistency (ACID): not critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.



Abstract Code

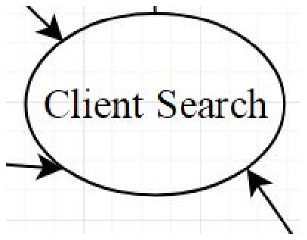
- User populates the *username* ('$Username'), *password* ('$Password') input fields.
- When ***Login*** button is clicked:
  - If the *username* or *password* field is empty:
    - Stay in **Login** form, with the message "Username and password are required. Please try again"
  - If both *username*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

Task Decomposition

Lock Types: Read-only on Client table
Number of Locks: Single
Enabling Conditions: Triggered by clicking the ***Client Search*** button on **User Home** form

Frequency: Frequent
Consistency (ACID): not critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.



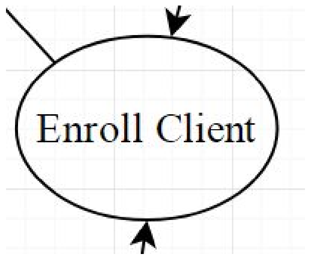
Abstract Code

- User populates the *clientName* ('$ClientName) and/or *description* ('$Description') input fields.
- When ***Search*** button is clicked:
  - If both *clientName* and *description* are empty:
    - Stay in **Client Search** form, with the message "Client name or description must be populated, please try again."
  - If *clientName* and/or *description* are populated:
    - Lookup the Client table based on the search criteria(s) provided.
    - 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 '$ClientSearchResults'.
      - Go to **Client Search** form to display the search results
- When ***Detail*** button is clicked (note that ***Detail*** button is enabled only when a Client record has been selected):
  - Retrieve the Client record from the '$ClientSearchResults' session variable.
  - Lookup all modifications to Client records.
  - Lookup all services provided to Client.
  - Store modifications and services provided in the '$ClientData' session variables.
  - Go to **Client Report** form.
- When ***Search*** button is clicked:
  - Go to **Enroll Client** form. No data is required beforehand to enroll a new client.

Enroll Client

Task Decomposition

Lock Types: Insert on Client table
Number of Locks: Single
Enabling Conditions: Triggered by selecting ***Enroll Client*** on the **Client Search** Form
Frequency:
Consistency (ACID): not critical, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.



Abstract Code

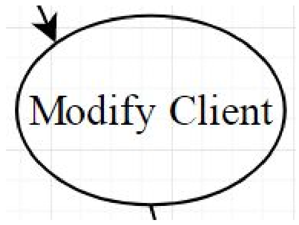
- User enters *clientName* (‘$ClientName), *description* (‘$Description), and/or *phoneNumber* (‘$PhoneNumber’) input fields.
- When the **Enroll Client** button is clicked:
    - If *clientName* or *description* field is empty:
        - Stay in **Enroll Client** form, with a message of both client name and description must be provided
    - If both *clientName* and *description* are provided:
        - Lookup *clientName* and *description* values on the Client table
        - If one or more Client records exist with *clientName* and *description* values:
            - Go back to **Enroll Client** form, with a message of Client already exists.
        - Else
            - Create a new Client record with *clientName, description,* and *phoneNumber*.
            - Store the Client information as a session variable ‘$ClientData’.
            - Go to **Client Report** form.

Modify Client

Task Decomposition

Lock Types: Select and Update on Client table
Number of Locks: Single

Enabling Conditions: Triggered by clicking the Update button on the Client Report form
Frequency: Infrequent
Consistency (ACID): Update must be completed in one step, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.



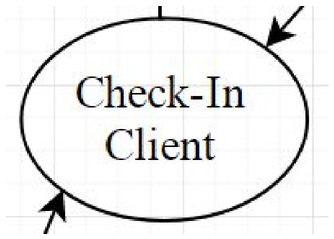
Abstract Code

- User modifies *clientName* ('$ClientName) and/or *description* ('$Description') and/or *phoneNumber* ('$PhoneNumber') fields in **Client Report** form.
- When the **Update** button is clicked:
    - If *clientName* or *description* field is empty:
        - Stay in **Client Report** form, with a message of both client name and description must not be empty.
    - If both *clientName* and *description* are populated:
        - Update Client table with the *clientName*, *description,* and *phoneNumber* values using *clientId* from the '$ClientData' session variable.
        - If update was successful:
            - Lookup Client record using the *clientId* from the '$ClientData' session variable.
            - Store Client record into the '$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

Task Decomposition

Lock Types: Select and Insert on ClientLog table
Number of Locks: Single
Enabling Conditions: Triggered by clicking the ***Check-In Client*** button on the Client Report form
Frequency: Frequent
Consistency (ACID): Update must be completed in one step, order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.



Abstract Code

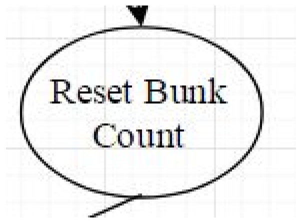
- User populates *datetime* ('$Datetime) and *description* ('$Description') and/or *note* ('$Note') fields in **Check-In Client** form.
- When the **Save** button is clicked:
  - If *datetime* or *description* field is empty:
    - Stay in **Check-In Client** form, with the message "Date/time and description are required, please try again.".
  - If both *datetime* and *description* are populated:
    - Insert into ClientLog table with the *datetime*, *description,* and *note* values using *clientId* and *site* values stored session variables.
    - If update was successful:
      - Lookup Client record using the *clientId* from session variable.
      - Lookup all modifications to Client records.
      - Lookup all services provided to Client.
      - Store Client data, modifications and services provided in the '$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

Task Decomposition

Lookups of shelter table, site form and updates bunk information in site form.
Lock Types: Read and update
Number of Locks: Multiple
Enabling Conditions: User login and edit request
Frequency: High frequency
Consistency (ACID): The lookup for shelter table need to be done first followed by information updating in site form.

Subtasks: Should be decomposed into subtasks. Mother task is needed.



Abstract Code

- View site information from **<u>Site Form</u>**.
- 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 **<u>Site Form</u>** and save all the information
    - Go back to site information page
  - If ***Cancel*** button is pushed:
    - Go back to site information page

<u>Report Available Bunks</u>

Task Decomposition

Two lookups of site and shelter information including the number of male/female/mixed bunks available.
Lock Types: Read-only
Number of Locks: Multiple
Enabling Conditions: No user authentication
Frequency: High frequency
Consistency (ACID): The lookup for site information need to be done first in order to find associated shelter service.
Subtasks: No mother task is needed.

Report Available Bunks

Abstract Code

- Iterate through Site table
- If this site has the shelter service, then:
    - Go to Service/Shelter table
    - If zero availability of bunks in this shelter:
        - Go back to Site table
    - Else:
        - Display shelter name ('$FacilityName'), site location('$location'), phone number('$phone'), hours of operation ('$HoursOfOperation') and conditions('$EligibilityCondition'), as well as the number of male/female/mixed bunks available('$BunkCount') in the **Site Report** Form
        - Go to **Bunk Report** Form

Report Meals Remaining

Task Decomposition

Three lookups of site, food bank and item/food information in order to retrieve the number of foods remaining in each food category.
Lock Types: Read only
Number of Locks: Multiple
Enabling Conditions: No user authentication
Frequency: High frequency
Consistency (ACID): The lookup for site information need to be done first in order to find associated food bank service.
Subtasks: No mother task is needed.

Abstract Code

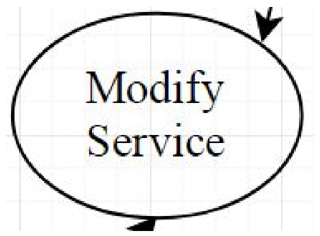
- Iterate through Site table
- If this site has the food bank service, then:
    - Go to Service/food bank table
    - Find FacilityName ('$FacilityName') of the food bank and save it as a key in **Meal Report** Form

- Find AvailableQuantity ('$AvailableQuantity') of the food in each FoodCategory ('$FoodCategory') and save them in Meal Report Form
- Sum up all the numbers of food in all the food banks under different FoodCategory ('$FoodCategory')
- Sum up total number of Meat/seafood and Dairy/eggs as one category
- Compare the total numbers of three food category Vegetable, nuts/grains/beans, and Meat/seafood OR Dairy/eggs and find the minimum of the three
- Display the minimum number as the number of the meals remaining in inventory in all food banks in the ASA system
- Display the FoodCategory ('$FoodCategory') which has the minimum number as type of donations are most needed to provide more meals

Modify Service

Task Decomposition

Lookups of site information.
Lookups of service lists.
Edits of site information and service information.
Lock Types: Read, insert and update
Number of Locks: Multiple
Enabling Conditions: User login and edit request
Frequency: Different frequencies of subtasks
Consistency (ACID): The lookup for site information need to be done first followed by lookup of service lists.
Subtasks: Should be decomposed into subtasks. Mother task is needed.



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:

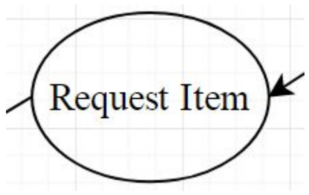
- Insert the service into ServiceList (‘$ServiceList’) of the site
- Display the **Service-Info** form with the selected service
- If ***Save*** button is pushed:
    - Go to Service table and save all the information
    - Go back to site information page
- If ***Cancel*** button is pushed:
    - Go back to site information page

- 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
        - 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:
            - Go to Service table and lookup the related service information
            - Delete all the related information
            - Go back to site information page
        - If ***Cancel*** button is pushed:
            - Go back to site information page

Request Item

Task Decomp

Lock Types: Write-only lock on Request table.
Number of Locks: Single
Enabling Conditions: Triggered when user requests an item.
Frequency: low frequency.
Consistency (ACID): Not Critical.
Subtasks: mother task or subtask is not needed.



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.

Item Search

Task Decomp

Lock Types: Read lock on Food Bank table when viewing the item report, Write Lock on Food Bank table when modifying the number of the item.
Number of Locks: Single
Enabling Conditions: Triggered when user search an item.
Frequency: low frequency.
Consistency (ACID): Critical, need to modify the number of items.
Subtasks: mother task or subtask is not needed.

Item Search

Abstract Code

- Click ***Search Item*** button from **Item Search** form. Jump to **Search Item** task.

- 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.
- 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)

<u>Retrieve Outstanding Requests</u>

Task Decomp

Lock Types: Read-only lock on Request table.
Number of Locks: Single lock for viewing the same table for different sorting.
Enabling Conditions: Triggered when user views outstanding request report.
Frequency: low frequency.
Consistency (ACID): Not Critical.
Subtasks: mother task is View Outstanding Requests; subtasks are viewing the report Sorted by StorageType ('$StorageType'), Sorted by Category ('$Category'), Sorted by Subcategory ('$SubCategory'), Sorted by Quantity of Items ('$Quantity').

Abstract Code

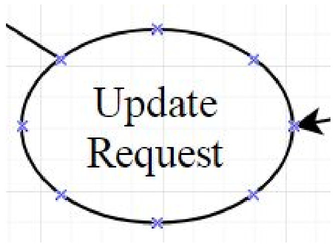
- Click ***Retrieve Outstanding Requests*** button from **<u>Site</u>** form. Jump to **Retrieve Outstanding Requests** task to show **<u>Outstanding Request Form</u>**.
    - By default, result in **<u>Outstanding Requests</u>** Form is sorted by '$StorageType'.
- Click ***Sorted by Storage Type*** button to view the report Sorted by '$StorageType'.
- Click ***Sorted by Category button*** to view the report Sorted by '$Category'.
- Click ***Sorted by Subcategory*** button to view the report Sorted by '$SubCategory'.
- Click ***Sorted by Quantity of Items*** button to view the report Sorted by '$Quantity'.
- 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.

Update Request

Task Decomp

Lock Types: Write-only lock on Request table.
Number of Locks: Single
Enabling Conditions: Triggered when user updates the request from **Outstanding Requests** Form.
Frequency: low frequency.
Consistency (ACID): Need consistency.
Subtasks: mother task or subtask is not needed.



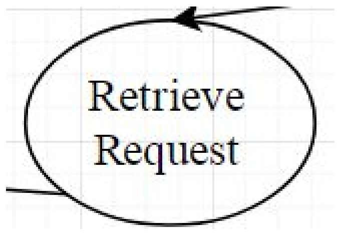
Abstract Code

- 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 ('$FulfilledNumber') to whatever request number is made.
   - If ***Reduce Request*** button is selected, look up that Request in **Requests** Form update '$FulfilledNumber' to the fulfillment number that user typed in.

Retrieve Request

Task Decomp

Lock Types: Read-only lock on Request table.
Number of Locks: Single
Enabling Conditions: Triggered when user retrieves all requests status.
Frequency: low frequency.
Consistency (ACID): Not critical
Subtasks: mother task or subtask is not needed.



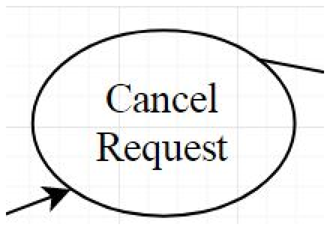
Abstract Code

- Click ***Retrieve Requests Status*** button. Jump to **Retrieve Requests Status** task.
- Look up Request table and return status (pending or closed) attribute.
  - If status is closed, return actual number provided.

Cancel Request

Task Decomp

Lock Types: Write-only lock on Request table.
Number of Locks: Single, user can only cancel his/her own request.
Enabling Conditions: Triggered when user cancels the request.
Frequency: low frequency.
Consistency (ACID): Critical, needs consistency.
Subtasks: mother task or subtask is not needed.



Abstract Code

- Click ***Cancel Request*** button from **<u>View Request Status</u>** 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.