Phase I Report
CS6400 – Spring 2017
Team 019

TABLE OF CONTENTS

Phase I Report
CS6400 – Spring 2017
Team 019

SQL DATA TYPES

Table : SITES

email ID	varchar(250) NOT NULL
short Name	varchar(50) NOT NULL
Phone Number	int(12) unsigned NOT NULL
Physical location - zip code	int(6) unsigned NOT NULL
Physical location - city	varchar(50) NOT NULL
Physical location - state	varchar(40) NOT NULL
Physical location - street Address	varchar(100) NOT NULL

Table : SERVICE_FOODBANK

status	ENUM(ACTIVE/INACTIVE)
Site email	Varchar(250) NOT NULL

Table : SERVICE_SHELTERS

Hours of Operation	varchar(100) NOT NULL
Conditions	varchar(500)
Description	varchar(500)
Category	ENUM(MALE, FEMALE, MIXED)
number available	Int(4) NOT NULL
Rooms available	Int(4) NOT NULL
status	ENUM(ACTIVE/INACTIVE)
Site email	varchar(250) NOT NULL

Table : SERVICE_FOODPANTRY

Hours of Operation	varchar(100) NOT NULL
Conditions	varchar(500)
Description	varchar(500)
status	ENUM(ACTIVE/INACTIVE)
Site email	Varchar(250) NOT NULL

Table : SERVICE_SOUPKITCHEN

Hours of Operation	varchar(100) NOT NULL

Conditions	varchar(500)
Description	varchar(500)
Seats available	int(4) NOT NULL
status	ENUM(ACTIVE/INACTIVE)
Site email	Varchar(250) NOT NULL

Table : USERS

First name	varchar(100) NOT NULL
Last name	varchar(100) NOT NULL
password	varchar(50) NOT NULL
email ID	varchar(250) NOT NULL
username	Varchar(25) NOT NULL
Site email	Varchar(250) NOT NULL

Table : REQUESTS

status	ENUM(PENDING, APPROVED)
# of units	Int(4) NOT NULL
username	Varchar(25) NOT NULL
dest site email	Varchar(250) NOT NULL
Time of request	TIMESTAMP() NOT NULL
Items name	Varchar(40) NOT NULL
Item sub-category	ENUM(PERSONAL_HYGINE, CLOTHING, SHELTER, OTHER) or ENUM(VEGETABLES, NUTS/GRAINS/BEANS, MEAT/SEAFOOD, DAIRY/EGGS, SAUCE/CONDIMENT/SEASONING, JUICE/DRINK)
Item expiry date	DATE() NOT NULL default 01/01/9999

Table : ITEMS

sub-category	ENUM(PERSONAL_HYGINE, CLOTHING, SHELTER, OTHER) or ENUM(VEGETABLES, NUTS/GRAINS/BEANS, MEAT/SEAFOOD, DAIRY/EGGS, SAUCE/CONDIMENT/SEASONING, JUICE/DRINK)
expiry date	DATE() NOT NULL
# of units	int(16) unsigned NOT NULL
Storage Type	ENUM(DRYGOOD, REFRIGERAATED, FROZEN)
Category	ENUM(FOOD, SUPPLIES)
Item Name	varchar(40) NOT NULL
Site email	Varchar(250) NOT NULL

Table: CLIENTS

First Name	varchar(100) NOT NULL
last Name	varchar(100)
Phone Number	int(12) may be NULL
ID Proof Number	Int(50) NOT NULL
ID Proof Description	varchar(100) NOT NULL

Table: WAITLIST

waitlistNumber	int($) NOT NULL
Site email	Varchar(250) NOT NULL
Client ID proof number	Int(50) NOT NULL
Client ID proof description	Varchar(100) NOT NULL

Table : CLIENT_LOGS

Client ID proof number	Int(50) NOT NULL
Client ID proof description	Varchar(100) NOT NULL
log Description	TIMESTAMP() + varchar(100)

CONSTRAINTS

1. User cannot delete all services from a site, *i.e*, a site should have at least one service.

2. Each site must be associated to at least one user.

3. User should be associated with only one site.

4. Clients cannot be deleted by users. All services used by the client are logged.

5. When a Client (head of household) is removed from waitlist to be given a room, user must explicitly add a log entry.

6. Available bunks/rooms and Meals Remaining Reports are accessible without user authentication.

7. Users when searching for a client, should enter the ID/name long enough such that it does not match more than 5 entries in the database.

8. A user cannot request items from his own site.

9. Users of one site should only be able to view the length of wait list at other sites.

LEGEND

Bold Italics → Button

Bold Underline → Document

$DollarName → variable

Bold → task

UPPERCASE → table name

VIEW WAITLIST

Task Decomposition:

Lock Types: Read-only on WAITLIST table

Enabling Conditions: Successful login followed by a click on ***View Waitlist***

Frequency: very high

Schemas: Single

Consistency: not critical, ordering is needed.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User selects ***View Waitlist*** from **User Main Menu**
- Get the site information that this user has admin-control on.
- Lookup WAITLIST table for all the entries matching that site.
- Display the entries in ascending order of WaitlistNumber.
- Upon
  - Click ***Edit*** button - run **Edit Waitlist.**
  - Click ***Close*** button - go to **User Main Menu** and run **Display User Main Menu.**

Task Decomposition:

Lock Types: Read-write on WAITLIST table

Enabling Conditions: Trigger by ***Edit*** from **Waitlist Form.**

Frequency: medium

Schemas: Single

Consistency: not critical, ordering is needed. .

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User selects ***Edit*** from **Waitlist Form**.
- Run **View Waitlist** task**.** Add two more input boxes to take *old waitlist number* ($owl) and *new waitlist number* ($nwl) as inputs. Let us call current waitlist number for each entry in the table ($cwl)
- Validate the input values are integers. Modify only those entires that belong to site which user has admin-control of.
- Click ***delete –*** If second box ($nwl) is filled, throw an error "Enter only one input in first box"

  Else:

  delete the entry for which Site-email=$Site-email and waitlistnumber=$owl

  For all entries where Site-email=$Site-email and WLnumber > $owl, decrement Wlnumber

  User has to manually update available rooms number as this is a rare scenario.
- Click ***change*** -
  - Throw Error if the given $cwl and $nwl are not positive integers or if any one of them is greater than length of the waitlist displayed.
  - Read the entry where Site-email=$Site-email && $WaitlistNumber = $owl
  - If $nwl < $owl :

    For all entries where (Site-email=$Sitee-email && $cwl >= $nwl && $cwl < $owl)

    Increment $cwl

    else :

    For all entries where (Site-email=$Site-email && $cwl >= $owl && $cwl < $nwl

    decrement $cwl
  - Insert the entry with WaitlistNumber=($nwl)
  - Run **View Waitlist** task.
    - Click ***Close*** button -go to **User Main Menu** and run **Display User Main Menu.**

ADD HEAD OF HOUSEHOLD TO WAITLIST

Task Decomposition:

Lock Types: Read-write on WAITLIST table

Enabling Conditions: Trigger by *Add to waitlist* from **Client report**.

Frequency: Medium

Schemas: Single

Consistency: critical, ordering is needed.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User selects ***Add to Waitlist*** from **Client report**.
- Get the site which the user has admin-control of, and clients ID proof num and type from **Client report**
- Look for a matching entry with site email and ClientID num and type. If it exists, throw an error and stay in **Client report**
- Else, get the $Count of entries from WAITLIST table matching $Site-email.
- INSERT on WAITLIST table with site email, ClientID num, type and $WaitlistNumber= ($Count + 1)
- Run **Get Client Logs and Data.**

CLIENT SEARCH

Task Decomposition:

Lock Types: Read-only on CLIENTS table

Enabling Conditions: Trigger by "***search by ID"*** *or "****search by name***" from **Client Search**.

Frequency: very high

Schemas: Single

Consistency: critical, ordering is needed. .

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User enters search text and should select one of ***search by name*** or ***search by ID***.
- If ***search by name***

  Lookup CLIENTS table with partial match on Name attribute

  Else

  Lookup CLIENTS table with partial match on ID Proof Number attribute
- Get the count of matching entries.
- If count = 0

  Display "no clients found with matching attributes".

  Else if count <= 5

  Display matching entries

  Else

  Display a message asking User to enter a longer search string.
- Upon selecting one of the Client, go to **Client Report** and run **Get Client Logs and Data** with the $Clientid.

ADD OR DELETE SERVICES

Task Decomposition:

Lock Types: Read-write on all SERVICE_* tables

Enabling Conditions: On a successful login, user can choose to add/delete services for his site.

Frequency: Low

Schemas: Multiple

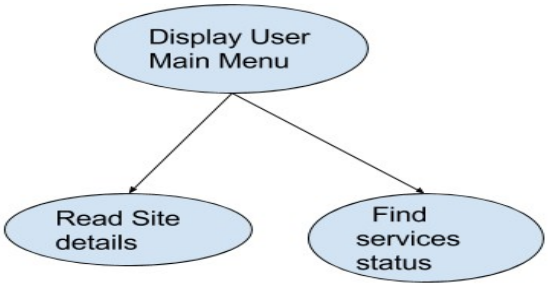
Consistency: critical, ordering is needed.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- **Display User Main Menu** displays the list of services that are active and inactive with a ***Delete*** button for active tasks and ***Add*** button for inactive tasks.
- If only one service is active, then disable the ***Delete*** button next to it.
- Get the details of the logged in user and the site which he has admin-control of .
- Get the service which was selected and the new status.
- Depending on the service selected go to the corresponding table. SERVICE_FOODPANTRY for food pantry, SERVICE_SOUPKITCHEN for soup kitchen, SERVICE_SHELTERS for shelters, and SERVICE_FOODBANK for food banks.
- Find the entry matching the $Site-email and change the $Status attribute based on input.
- Run **Display User Main Menu.**

DISPLAY USER MAIN MENU



Task Decomposition:

Lock Types: Read-only on USERS, SITES and all SERVICES_* tables

Enabling Conditions: Triggered on successful login.

Frequency: very high

Schemas: Multiple

Consistency: not critical, ordering not needed.

Subtasks: Decomposition needed as multiple tables are involved.

Abstract Code:

- Upon successful login, get the User details from USERS table using $Username
- Get the $Site-email by reading USERS table.
- Find the Site details from SITES table using Site-email and display them.
- Find the status of all services by reading the entry corresponding from each of SERVICE_FOODPANTRY, SERVICE_SOUPKITCHEN, SERVICE_SHELTERS, SERVICE_FOODBANK and display the services in active and inactive categories respectively with ***Add/Delete*** buttons next to them.
- Upon
  - Click ***View Inventory*** - go to **Food Bank Status Form** and run **View Current Inventory**.
  - Click ***Item Search*** - go to **Item Search Form**
  - Click ***Add*** / ***Delete*** -run **Add Or Delete Services**
  - Click ***Add Client*** - go to **Add Client Form** and run **Add Client**
  - Click ***Add Items –*** go to **Add Items Form**
  - Click displayed services to edit attributes of corresponding service.
  - Click ***Search Clients*** - go to **Client Search Form**
  - Click ***View Waitlist*** - go to **Waitlist form**
  - Click ***Logout*** - run **Logout** and go to **Global Reports Form**

LOGIN

Task Decomposition:

Lock Types: Read-only on USERS table

Enabling Conditions: on selecting login from **Global Reports form**.

Frequency: High

Schemas: Single

Consistency: not critical, ordering not needed.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User enters *username* ($Username) and *password* ($Password) fields.
- If ***login*** is selected then
    - Validate data for both user *username* and *password* fields
    - Search USERS table for the $Username provided, get the entry details.
    - If entry is found and password = $Password

        Store login information as session variable $Username.

      Else:

        Display error message and redisplay **Login form**.
- If ***cancel*** is selected go to **Global Reports Form**

ADD CLIENT

Task Decomposition:

Lock Types: Read-write on CLIENTS table

Enabling Conditions: None

Frequency: low

Schemas: Single

Consistency: not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:

- User clicks on ***Add Client*** from **User Main Menu**
- The user has to enter Client First Name, Last name, ID/Description, Phone # (Optional) as the input fields.
- Validate,
    - First Name, Last Name, ID and Description are valid varchar fields and within the character limits of their respective fields. Throw error otherwise
    - Phone number is a number of 10 digits or NULL
- Upon
    - Click ***Submit***
        - Validate mandatory fields and all input data, throw an error message if not valid.
        - Look up for the same client in the database using ID/Description as key.
        - If client exists

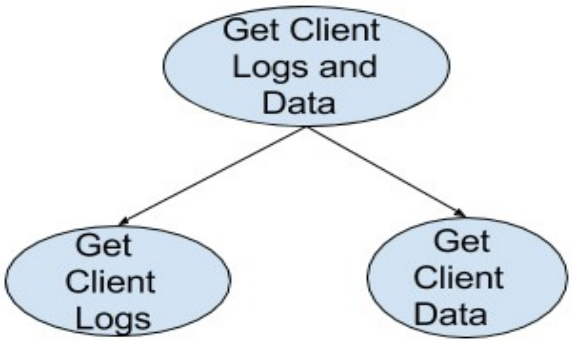
            throw error

          Else

            perform an insert in CLIENTS table with the given inputs and

            Goto **User Main Menu** and run **Display User Main Menu**.
    - Click ***Cancel*** - Jumps to **User Main Menu** without making any Update.

GET CLIENT LOGS AND DATA



Task decomposition:

Lock Types: Read-only on CLIENT_LOGS table and CLIENTS Table

Enabling Conditions: None

Frequency: very high

Schemas: Multiple

Consistency: not critical, order is not critical.

Task Decomposition: Can read tables in paralled. Can be divided into two subtasks.

Abstract code:

- The User can view Client Report when he selects a particular Client from Client Search.
- Once the User selects a particular Client, get the Client's ID number and type.
- Read the client's details from CLIENTS table by $ClientIDNum and $ClientIDType and populate report.
- Read the CLIENT_LOGS table and get all entries which match with the Client details and sort them in descending order with timestamp.
- Upon
    - Click ***Add To Waitlist*** - run **Add Head Of Household To Waitlist.**
    - Click ***Edit Client*** - run **Edit Client**
    - Click ***Add Service log*** - run **Add log**

EDIT CLIENT

Task Decomposition:

Lock Types: Read-write on CLIENTS table

Enabling Conditions: triggered by selecting Edit Client from **Client Report**

Frequency: Low

Schemas: Single

Consistency: not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract code:

User clicks on ***Edit client*** button from the **Client Report**.

- Client *First name*, *Last Name*, *ID Number, ID Proof Description*, *Phone Number fields* needs to be retrieved by using a Select on the CLIENTS table.
- The user can Edit any of the given fields and Submit.
- Upon
  - Click ***Submit***:

    ➔ Validate the input

    ➔ Show appropriate error message if any validation fails

    ➔ If input is valid, update the CLIENTS table with modified client information.
  - Click ***Cancel*** - Jumps to **User Main Menu** without making any update.
- Run **Get Client Logs and Data**.

ADD LOG

Task Decomposition:

Lock Types: Read-write on CLIENT_LOGS table and SERVICE_SHELTERS table.

Enabling Conditions: triggered by selecting Add Log from **Client Report**

Frequency: very High

Schemas: Single

Consistency: not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract code:

- User clicks on ***Add Log*** button from the **Client Report**.
- Take the description entered by User as input $Log
- Validate that the enter value of $Log is varchar and within the character limit of the field definition. Throw error otherwise.
- Display three check-boxes to mark if it is bunk check-in one for each Male bunk or Female bunk or a Mixed bunk.
- On Click ***Save***: Validate the input and append it with ClientID details and $Timestamp and insert into CLIENT_LOGS table and run **Get Client Logs and Data**.
- Decrement the number of bunks of the particular type for this site by one if any checkbox is marked.
- On Click ***Cancel*** - Jumps to **User Main Menu** without making any update.

LOGOUT

Task Decomposition:

Lock Types: None

Enabling Conditions: triggered by selecting Logout from **User Main Menu**

Frequency: High

Schemas: none

Consistency: not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract code:

- User clicks on ***Logout*** button from the **User Main Menu**.
- Invalidate the session and go to **Global Reports Form.**

VIEW CURRENT INVENTORY

Task Decomposition:

Lock types: Read lock on the ITEMS table

Enabling conditions: Valid Login followed by click on ***View Current Inventory***

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Lookup ITEMS table based on the current $Site-email
- Display $NumUnits $Sub-category $expiry and $ItemName for each item in the ITEMS table.
- Upon:
    - Click ***Back*** button - Go back to **User Main Menu** .

VIEW NUMBER OF SEATS

Task Decomposition:

Lock types: Read lock on SERVICE_SOUPKITCHEN table

Enabling conditions: Valid login followed by click on ***Soupkitchen*** on **User MainMenu**

Frequency: Low

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Lookup SERVICE_SOUPKITCHEN based on $Site-email
- Display $SeatsAvailable from SERVICE_SOUPKITCHEN Table and active status.
- Upon:
    - Click ***Back*** button - Go back to **User Main Menu** task.
    - Click ***Edit*** button - Run **Edit Seats Available** task

EDIT NUMBER OF SEATS

Task Decomposition:

Lock Types: Read-Write on SERVICE_SOUPKITCHEN table.

Enabling Conditions: Triggered by selecting ***Edit*** button on **Soup Kitchen Form**.

Frequency: very Low

Schemas: Single

Consistency: not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract code:

- User clicks on ***Edit*** button from **Soup Kitchen Form**
- Read the new $NewSeatCount from input. Validate that the entered value of $NewSeatCount is a valid integer and non-negative.
- Upon
    - Click ***Submit*** - Update the $SeatCount field to $NewSeatCount on SERVICE_SOUPKITCHEN table for the $Site-email.
    - Click ***Cancel*** - Jumps to **Soup Kitchen Form** without making any Update.

VIEW NUMBER OF BUNKS AND ROOMS

Task Decomposition:

Lock types: Read lock on the SERVICE_SHELTERS Table

Enabling conditions: Valid login followed by click on ***Shelters***

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Lookup SERVICE_SHELTERS based on $Site-email
- Display $NumberOfBunks, $Site-email from SERVICE_SHELTERS Table.
- Upon:
    - Click ***Back*** button - Go back to **User Main Menu**.
    - Click ***Edit*** button - Jump to **Edit Number Of Bunks And Rooms** task

EDIT NUMBER OF BUNKS AND ROOMS

Task Decomposition:

Lock types: Read-write lock on the SERVICE_SHELTERS Table

Enabling conditions: Triggered by selecting ***Edit*** button on **Shelters Form**.

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- User clicks on ***Edit*** button from **Shelters Form**
- Read the new number of bunks $NewNumBunks and $NewNumRooms from input.
- Validate that the entered value of $NewNumBunks and/or $NewNumRooms is a valid integer and non-negative.
- Upon
  - Click ***Submit*** - Update on SERVICE_SHELTERS table for the $Site-email with the values of $NumBunks and $NumRooms updated to $NewNumBunks and $NewNumRooms respectively.
  - Click ***Cancel*** - Jumps to **Shelters form** without making any Update.

VIEW ALL INCOMING REQUESTS

Task Decomposition:

Lock types: Read lock on the REQUESTS Table

Enabling conditions: Valid Login followed by click on ***View All Incoming Requests List***

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Run the ***View All Incoming Requests List*** task by $Site-email
- Display all Item details and $Status from REQUESTS Table.
- Upon:
    - Click ***Back*** button - Go back to **User Main Menu**.
    - Click ***Accept/Modify*** button - Jump to **Accept/Modify Incoming Requests** task

ACCEPT OR MODIFY INCOMING REQUESTS

Task Decomposition:

Lock types: Read-write lock on the REQUESTS Table and ITEMS table

Enabling conditions: Valid Login followed by click on ***Accept/Modify Incoming Requests List***

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Read all incoming requests based on $Site-email from REQUESTS table.
- Display Item details and $Status from REQUESTS Table with one column each for an ***Accept, Modify*** and a ***Decline*** button.
- Disable the ***Accept*** button for all cases where the $NumUnits in REQUESTS table is greater than $NumUnits in ITEMS table for the same the same item.
- Upon:
    - Click ***Accept*** button:
        - ➔ Subtract the value of $NumUnitsReq from the REQUESTS table from the value of $NumUnits from the ITEMS table and store $NumUnits into the ITEMS table for the given item. Delete the record if $NumUnits is zero.
        - ➔ Change $Status in the REQUESTS table to “Closed” for the accepted request.
    - Click ***Modify*** button:
        - ➔ Make the number of Items editable.
        - ➔ Take the number of items ($Num) as input. Subtract $Num from $NumUnits and save the value to ITEMS table. Delete the record if $NumUnits is zero.
        - ➔ Change $Status in the REQUESTS table to “Closed” and update the number of units to $Num
    - Click ***Decline*** button:
        - Change $Status in the REQUESTS table to “Closed” for the declined request and update the number of items to zero for the corresponding entry in REQUESTS table.
    - Click ***Back*** button - Jump to **View All Incoming Requests List** task

ITEM SEARCH

Task Decomposition:

Lock types: Read/write lock on ITEMS Table

Enabling conditions: Valid Login followed by click on ***Item Search***

Frequency: Very High

Schemas: Single

Consistency: is not critical,even if other request are being made for that item

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Run the ***Item Search*** task based on $Site-email, $ItemQuery, $ExpiryDate, $StorageType $Category, $SubCategory, $User->Site-email and $DescQuery
- When ***Search*** button is clicked we run the item search task with all the parameters.
- IF $NumUnits <= 0 Delete item and update ITEMS table
- Display all the items with $Site->ShortName, $ItemName, $StorageType, $Category, $SubCategory, $Description, and $NumUnits.
- ***Request*** button will not appear against items where $User→Site-email = $Site-email. User can edit the $NumCount Value of Item here and click ***Save*** to save the $NumCount value in the ITEMS table
- Upon:
    - Click on ***Request*** button. -
        - run **Request Items** task.
    - Click ***Clear*** button - Jump to new **Item Search** task

GENERATE AVAILABLE BUNKS AND ROOMS REPORT

Task Decomposition:

Lock types: Read locks on SERVICE_SHELTER table, SITES table.

Enabling conditions: Triggerd by click on ***Generate Available Bunks And Rooms Report***

Frequency: High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Run the **Generate Available Bunks And Rooms Report** task
- Display for all $Site-email, the $SiteShortName, $SitePhysicalLocation, $SitePhoneNumber from SITES table and $Category, $HoursOfOperation, $ConditionsOfUse and $bunks_avialable and $rooms_availabe from SERVICE_SHELTERS Table.
- If no vacanat rooms and bunks are found, Display - “Sorry, all shelters are currently at maximum capacity.”
- Upon
    - Click ***Back*** button - Go back to **Global Reports Form** .

GENERATE AVAILABLE MEALS REPORT

Task Decomposition:

Lock types: Read lock on ITEMS table.

Enabling conditions: Triggerd by click on ***Generate Available Meals Report***

Frequency: Very High

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Calculate $MaxMealsPossible
  1. Read all rows from ITEMS table where $Category is FOOD
  2. Calculate *minimum* ($MinItems) of the following. Don't count if $ItemExpiry is in the past.
     a) Rows in ITEMS where $SubCategory is Vegetable
     b) Rows in ITEMS where $SubCategory is nuts, beans or grains
     c) sum of the number of
        - Rows in ITEMS where $SubCategory is meat or sea food
        - Rows in ITEMS where $SubCategory is dairy or eggs
- Store $ItemName list in ascending order of $NumCount in $LeastItemSubCategory from ITEMS table
- Assign the value of $MinItems to $MaxMealsPossible
- Displays $SiteName, $MaxMealsPossible under "Meals Available" and $LeastItemSubCategory under "Donations Needed".
- Upon:
  - Click ***Back*** button - Go back to **User Main Menu** task.

CATEGORIZE AND INSERT ITEMS FROM DONATIONS

Task Decomposition:

Lock Types: Read-write on ITEMS table

Enabling Conditions: Successful login followed by a click on ***Add Items***

Frequency: High

Schemas: Single

Consistency: not critical, ordering is needed.

Subtasks: Mother Task is not needed. No decomposition needed

Abstract Code:

- User selects ***Add Items*** from **User Main Menu**
- Populate the drop down boxes for Item Category and subcategory
- From the **Add Items Form**, read all the Item attributes
- If $ItemExpiry is in the past, throw error.
- On Click ***Add*** –
  - Lookup ITEMS table for matching item with (sub-category, name, expiration date)

    If a match is found

    increment the number of items for that entry by given input.

    Else :

    add a new entry in ITEMS table, with the given attributes.
- On Click ***Close*** button - go to **User Main Menu** and run **Display User Main Menu.**

VIEW ALL REQUESTS OF A USER

Task Decompostion:

Lock types: Read lock on the REQUESTS table

Enabling conditions: Valid login followed by click on ***View My Requests***

Frequency: Medium

Schemas: Single

Consistency: Critical

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code:

- Lookup REQUESTS table for all entries where $Username matches with Current user's $Username.
- Display $Num, Item details and $Status from REQUESTS Table.
- Upon:
  - Click ***Back*** button - Go back to **User Main Menu** .

REQUEST ITEMS

Task Decomposition:

Lock types: Read lock on the REQUESTS table and ITEMS table

Enabling conditions: click on ***Request*** button from **Item Search Form**

Frequency: Medium

Schemas: Single

Consistency: Critical

S**ubtasks:** Mother task is not needed. No decomposition needed

Abstract Code:

- User clicks on ***Request*** button from **Item Search Form**
- Popup an input screen asking for the number of items to request - take the input as $Num
- Insert into REQUESTS table with $Site-email, Details of the Item requested like sub-category, name and expiry, $Username and $NumUnits
- stay on **Item Search Form**