Hemkraft

Table of Contents

Data Types

Surrogates are not included for Bathroom, Appliance, and HeatSource.

Household

Attribute	Data Type	Nullable
Email	String	Not Null
SquareFootage	Integer	Not Null
NumOfOccupant	Integer	Not Null
NumOfBedroom	Integer	Not Null
HouseholdType	Enum	Not Null

PhoneNumber

Attribute	Data Type	Nullable
TenDigit	String	Not Null
AreaCode	String	Not Null
SevenDigit	String	Not Null
PhoneType	Enum	Not Null

Bathroom

Attribute	Data Type	Nullable
NumOfSink	Integer	Not Null
NumOfCommode	Integer	Not Null
NumOfBidet	Integer	Not Null

HalfBathroom

Attribute	Data Type	Nullable
Name	String	Null

FullBathroom

Attribute	Data Type	Nullable
NumOfBathtub	Integer	Not Null
NumOfShower	Integer	Not Null
NumOfTub	Integer	Not Null
IsPrimary	Boolean	Not Null

Location

Attribute	Data Type	Nullable
PostalCode	String	Not Null
City	String	Not Null

State	String	Not Null
Latitude	Double	Not Null
Longitude	Double	Not Null

Appliance

Attribute	**Data Type**	**Nullable**
ModelName	String	Null
ApplianceType	Enum	Not Null

Manufacturer

Attribute	**Data Type**	**Nullable**
ManufacturerName	String	Not Null

Refrigerator

Attribute	**Data Type**	**Nullable**
RefrigeratorType	Enum	Not Null

Washer

Attribute	**Data Type**	**Nullable**
LoadingType	Enum	Not Null

TV

Attribute	**Data Type**	**Nullable**
DisplayType	Enum	Not Null
DisplaySize	Enum	Not Null
MaxResolution	Enum	Not Null

Oven

Attribute	**Data Type**	**Nullable**
OvenType	Enum	Not Null

HeatSource

Attribute	**Data Type**	**Nullable**
HeatSourceType	Enum	Not Null

Business Logic Constraints

Household

- Anyone can submit household info and view the available reports of household information. Hemkraft does not require login or registration for users.
- Entering the household info associated with an existing email address will be prohibited: entering an existing email address will pop up an error message to alert the users.
- Users are not allowed to edit the existing household info.
- Users are not allowed to change data they have previously entered.
- The SquareFootage attribute represents the total square footage of the household, not the square footage of a portion of the household.
- The number of occupants should be the count of all occupants in the household, including any adults and children. Each child is counted one towards the number of occupants.
- A household could have no bedroom at all.

PhoneNumber

- Phone numbers will be used by Hemkraft staff to contact households only when household owners permit - users choose to enter phone number.
- A phone number cannot be shared with two or more households: trying to save an existing phone number to the database will result in an error and therefore entry will be denied.

FullBathroom

- A full bathroom can be deemed as the primary bathroom if it is attached to the primary bedroom. There can be as many as one primary bathroom, with the possibility of no primary bathroom.

HalfBathroom

- Hemkraft does not require the names of half bathrooms to be unique in the system.

Location

- Hemkraft only collects info of households from certain areas/regions. Users will be required to re-enter the postal code if a postal code outside of these areas is entered.
- All postal codes and the corresponding locations (i.e., states, cities) will never be changed in the system (database).

Appliance

- All possible manufacturers are pre-stored into the system (database)
- The list of manufacturers can be updated only by the database administrative staff.

- Only by choosing an appliance type can the user add the information of a new appliance.

Task Decomposition with Abstract Code: (TD/AC)

Format in Task Decomp and Abstract Code:
- Forms (Documents) -> **Bold Underline**
- Buttons/Links -> ***Bold Italics***
- Tasks -> **Bold**
- Input attributes in forms from EER -> *Italics*
- Entity -> Blue
- varaible's name in code -> 'email', 'postalCode', ...

Main Menu (not in IFD, no info flow in/out database)

Task Decomp

Lock Types: None

Number of Locks: None

Enabling Conditions: None

Frequency: High

Consistency(ACID): Consistency is not cirtial; Order is also not critical.

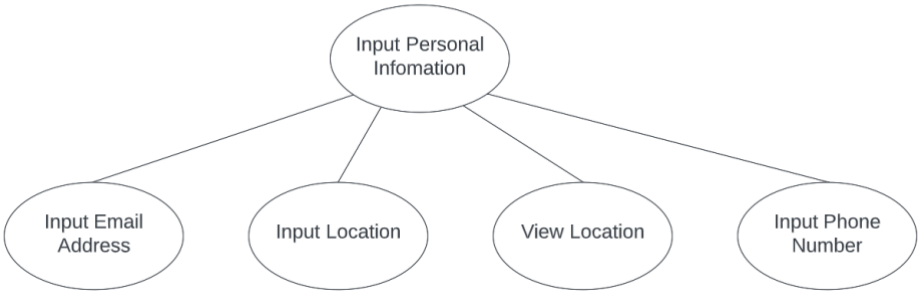
Subtasks: Mother task is not needed; No decomposition needed.

Abstract Code

- Show "***Enter my household info***" and "***View reports / query data***" buttons.
- Upon:
  - Click ***Enter my household info*** button - jump to the **Input Personal Information** task.
  - Click ***View reports / query data*** button - jump to the **View Reports** task.

Input Personal Information

Task Decomp



Lock Types:

Inserts of Email and PhoneNumber information for a Household;

Read-only look-up of City and State information in a Location;

Write of Location information for a Household.

Number of Locks:

3 locks

Enabling Conditions:

Input Personal Information task is triggered by clicking ***Enter my household info*** button in **Main Menu**.

Frequency:

Input Email Address, **Input Location** and **View Location** subtasks have same frequency;

Input Phone Number has a lower frequency because this subtask is not mandatory for user.

Consistency(ACID):

Consistency is not critical: even if the information is being retrieved to generate reports while the user is inserting it.

Subtasks:

Input Email Address, **Input Location** and **View Location** must be done.

Mother task is needed to coordinate subtasks and control the execution order.

Decomposition is needed.

Execution order of the subtasks:

1. First, **Input Email Address** subtask should be executed for user identification;
2. Then, **Input Location** and **View Location** can be executed.
3. At last, run **Input Phone Number**.

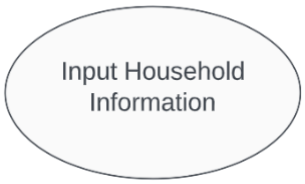
Abstract Code

- User clicked on ***Enter my household info*** button from **Main Menu**:
- Run the **Input Personal Information** task:
    - First display the **Email** form:
        - User enters *Email*('email').
        - If data validation is successful for *email* input field, then:
            - When the ***Submit*** button is clicked, insert the 'email' to database
                - If insertion to database failed (already exists in database):
                    - Go back to **Email** form, with an error message.
                - Else:
                    - Display **Location** form.
        - Else *email* input field is invalid, go back to **Email** Form, with an error message.
    - After **Location** form displayed:
        - User enters *PostalCode*('postalCode'): required to be 5 digits.
        - If data validation is successful for *PostalCode* input field, then:
            - When the ***Submit Postal Code*** button is clicked,
                - Find the Location instance using 'postalCode'.
                    - If no matched Location object is found in database, go back to **Location** form, with an error message.
                    - If matched Location object found, display this Location's *'postalCode'*, *'city'*, *'state'*.
                        - When the ***YES*** button is clicked, save Location information to current Household object, and go to **Phone Number** form.
                        - When the ***NO*** button is clicked, go back to **Location** form with an error message and let user re-enter *PostalCode*.
        - Else *PostalCode* input field is invalid, go back to **Location** form, with an error message.
    - After **Phone Number** form displayed:
        - If ***NO*** button is clicked, skip and go to the next task: **Input Household Information** task.
        - If ***YES*** button is clicked,

- User enters:
  - *AreaCode*('areaCode'): required to be 3 digits;
  - *SevenDigit* ('sevenDigit'): required to be 7 digits other than dash(dash is permitted);
  - *PhoneType*('phoneType'): for *PhoneType* field, users are only allowed to select one from the drop-down list which comprises four types {"home", "mobile", "work", "other"}.
- If data validation is successful for the input fields, then:
  - When the ***Next*** button is clicked, create a PhoneNumber object with the user inputs in, and insert it to database.
  - If insertion to database failed:
    - Go back to **Phone Number** Form, with an error message.
  - Else:
    - Save PhoneNumber information for current Household object, then run next **Input Household Information** task.
- Else any of the input fields are invalid, go back to **Phone Number** Form, with an error message.

Input Household Information

Task Decomp



Lock Types:

Write of Household information for a Household.

Number of Locks: 1 lock

Enabling Conditions:

Input Household Information task is triggered by the successful completion of **Input Personal Information** task

Frequency: High - All tasks have the same frequency

Consistency and Order (ACID): Consistency is not critical; Order is not critical.

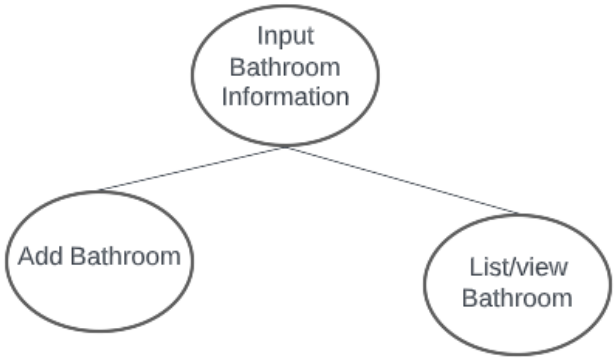
Subtasks: No mother task needed; decomposition is not needed; order is not necessary.

Abstract Code

- ***Next*** Button from previous page clicked and data collected from previous task passed validation:
    - Display **Household Info** form.
    - User enters:
        - *HomeType*('householdType'): user can select one of the *HomeType* from the dropdown list {"House", "Apartment", "Townhome", "Condominium", "Mobile Home"};
        - *SquareFootage*('squareFootage'): needs to be more than 0;
        - *NumOfOccupants*('numOfOccupant');
        - *NumOfBedroom*('numOfBedroom'): cannot be a negative number;
- When ***Next*** button clicked:
    - IF any properties are missing, display "Blanks Error" message to prompt user to complete all 4 fileds.
    - IF data validation is successful for all input fields,
        - save Household information collected in this task in the database.
        - go to next **Input Bathroom Information** task.
    - ELSE data validation for any input fields is invalid,
        - go back to **Household Info** form with an error message.

Input Bathroom Information

Task Decomp



Lock Types:

1 insert of sink, commode, bidet, and name information for HalfBathroom;

1 insert of sink, commode, bidet, bathtub, shower, tub, whether primary or not information for FullBathroom;

1 read-only look-up of Bathroom.

Number of Locks: 3 locks

Enabling Conditions:

Input Bathroom Information task is triggered by the successful completion of **Input Household Information** task;

Frequency: Low – all tasks have the same frequency

Consistency(ACID): not critical

Subtasks:

Mother task is required to coordinate subtasks.

Decomposition is required.

Order: **Add Bathroom** subtask should always be run before **List/View Bathroom** subtask.

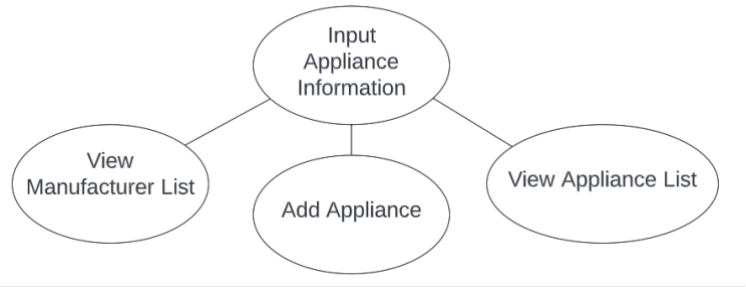
Abstract Code

- User clicks on ***Next*** button from previous page and data collected from previous task passed validation.
- Run **Add Bathroom**, then:
  - When ***Half*** button is selected:
    - Display a form for entering HalfBathroom information.
    - User enters information:
      - *NumberOfSink* (‘numOfSink’), *NumOfCommode* (‘numOfCommode’), *NumOfBidet* (‘numOfBidet’): the sum of these 3 input values should be larger than 0;
      - *Name* (‘name’).
  - When ***Full*** button is selected:
    - Display a form for entering FullBathroom information.
    - User enters information:
      - *NumberOfSink* (‘numOfSink’), *NumOfCommode* (‘numOfCommode’), *NumOfBidet* (‘numOfBidet’): no constraint for the sum.
      - *NumOfBathtub* (‘numOfBathtub’), *NumOfShower* (‘numOfShower’), NumOfTub (‘numOfTub’): the sum of these 3 input values should be larger than 0;
      - Whether the bathroom is a primary bathroom (‘isPrimary’):
        - If exists a primary FullBathroom object in database, user will not be able to check the current one to be primary.
  - User clicked on ***Add*** button:
    - If ***Half*** button is selected:

        - If data validation is successful for all input fields discussed above, then:
            - Save data to database for **HalfBathroom.**
            - Go to next **List/View Bathroom** subtask.
        - Else: go back for user to enter data again, with an error message.
    - Else:
        - If data validation is successful for all input fields discussed above, then:
            - Save data to database for **FullBathroom.**
            - Go to next **List/View Bathroom** subtask.
        - Else: go back for user to enter data again, with an error message.
- Then, run **List/View Bathroom** subtask:
    - Find all the saved bathrooms; Display each bathroom's type and 'isPrimary' values;
    - Upon clicking ***Add Another Bathroom*** button, go back and run **Input Bathroom Information** task.
    - Upon Clicking ***Next*** button, go to **Input Appliance Information** task.

Input Appliance Information

Task Decomp



Lock Types:

Insert of information for Appliance;

Read-only look-ups of Appliance information and Manufacturer list.

Number of Locks: 3 locks

Enabling Conditions:

Input Appliance Information is triggered by the successful completion of **Input Bathroom Information** task.

Frequency: High – this task is the main data collection task of the app.

Consistency(ACID):

Consistency is not critical: even if the information is being retrieved to generate reports/list all the appliance saved in database while the user is inserting it.

Subtasks:

Mother task is needed for coordination.

Decomposition is needed.

Execution order of the subtasks:

View Manufacturer List should always be run before **Add Appliance**.

View Appliance List should be run after **Add Appliance** (**Add Refrigerator/Cooker/Washer/Dryer/TV**).

Abstract Code

- Run **Input Appliance Information** task, display **Appliance** form;
- User is prompted to choose *ApplianceType*('applianceType') for the Appliance; User can only select one of the *ApplianceType* in a drop down menu includes {"Refrigerator/freezer", "Cooker", "Washer", "Dryer", "TV"}.
- After user selects *ApplianceType*, run **View Manufacturer List** subtask:
    - Find all the Manufacturer objects; Display manufacturers' *ManufacturerName* in a drop-down list.
    - User selects Manufacturer information for current Appliance.
- User can optionally enter *ModelName* ('modelName') for Appliance.
- When selecting *ApplianceType,*
    - If user chooses ***Cooker***, display a form for entering Cooker information.
        - User can click on ***Oven*** or ***Cooktop*** button to enter information, also can click both buttons to enter information for both.
            - If ***Oven*** button is clicked, user will be able enter information required by Oven:
                - *OvenType*('ovenType') is selected from a drop-down menu includes: {"Convection", "Conventional"};
                - HeatSource information is also entered by user. The *HeatSourceType*('heatSourceType') can only be chosen from {"Gas", "Electric", "Microwave"}, but the user can choose multiple of the types.
            - If ***Cooktop*** button is clicked, user will be able enter information required by Cooktop:
                - HeatSource information can be entered by user. The *HeatSourceType*('heatSourceType') can only be chosen

from {“Gas”, “Electric”, “Radiant electric”, “Induction”}. User can only choose one of them.

- If data validation is successful for all input fields discussed above, then:
  - When the ***Add*** button is clicked,
    - Save Cooker information to database.
    - Go to **View Appliance List** subtask.
- Else any input field is invalid, go back for user to enter Cooker information again, with an error message.

- If user chooses ***Refrigerator/freezer***, display a form for entering Refrigerator information.
  - User enters information required by Refrigerator:
    - *RefrigeratorType* (‘refrigeratorType’): user can choose one type from {“Bottom freezer refrigerator”, “French door refrigerator”, “Side-by-side refrigerator”, “Top freezer refrigerator”, “Chest freezer”, “Upright freezer”}.
  - If data validation is successful for input fields, then:
    - When the ***Add*** button is clicked,
      - Save Refrigerator information.
      - Go to **View Appliance List** subtask.
  - Else any input field is invalid, go back for user to enter Refrigerator information again, with an error message.
- If user chooses ***Washer***, display a form for entering Washer information.
  - User enters information required by Refrigerator:
    - *LoadingType*(‘loadingType’): choose one from {“Top”, “Front”}.
  - If data validation is successful for input fields, then:
    - When the ***Add*** button is clicked,
      - Save Washer information.
      - Go to **View Appliance List** subtask.
  - Else any input field is invalid, go back for user to enter Washer information again, with an error message.
- If user chooses ***Dryer***, display a form for entering Dryer information.
  - User enters information required by Dryer:
    - HeatSource information for Dryer is also entered by user. The *HeatSourceType*(‘heatSourceType’) can only be chosen from {“Gas”, “Electric”, “None”}; User can choose one.
  - If data validation is successful for input fields, then:
    - When the ***Add*** button is clicked,
      - Save Dryer information.
      - Go to **View Appliance List** subtask.
  - Else any input field is invalid, go back for user to enter Dryer information again, with an error message.
- If user chooses ***TV***, display a form for entering TV information.
  - User enters information required by TV:

        - *DisplayType*(‘displayType’): user can select one from {“Tube”, “DLP”, “Plasma”, “LCD”, “LED”};
        - *DisplaySize*(‘displaySize’): user can select one from {“Tube”, “DLP”, “Plasma”, “LCD”, “LED”};
        - *MaxResolution*(‘maxResolution’): user can select one from {“480i”, “576i”, “720p”, “1080i”, “1080p”, “1440p”, “2160p (4K)”, “4320p (8K)”}.
    - If data validation is successful for input fields, then:
        - When the ***Add*** button is clicked,
            - Save TV information.
            - Go to **View Appliance List** subtask.
    - Else any input field is invalid, go back for user to enter TV information again, with an error message.
- Then, run **View Appliance List** subtask:
    - Find all the saved appliances; Display each appliance’s ‘applianceType’, ‘manufacturer’, ‘modelName’;
    - Upon clicking ***Add Another Appliance*** button, go back and run **Input Appliance Information** task.
    - Upon Clicking ***Next*** button, go to **Wrap Up** task.

Wrap Up (not in IFD, no info flow in/out database)

Task Decomp

Lock Types: None

Number of Locks: None

Enabling Conditions: None

Frequency: high

Consistency(ACID): Consistency is not cirtial; Order is also not critical.

Subtasks: Mother task is not needed; No decomposition needed.

Abstract Code

- Show “***Reture to main menu***” button.
- Upon Click ***Reture to main menu*** button, jump to the **Main Menu** task.

View Reports (not in IFD, no info flow in/out database)

Task Decomp

Lock Types: None

Number of Locks: None

Enabling Conditions: None

Frequency: high

Consistency(ACID): Consistency is not cirtial; Order is also not critical.

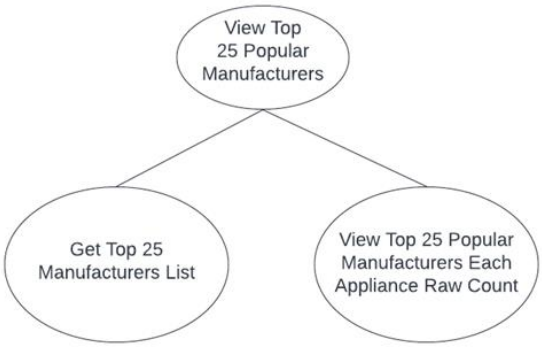
Subtasks: Mother task is not needed; No decomposition needed.

Abstract Code

- Show "***View Top 25 Popular Manufactures***", "***View Bathroom Statistics***", "***View Laundry Center Report***", "***View Extra Fridge/freezer Report***", "***View Average TV Display Size by State***", "***Search Manufacture / Model***", and "***View Household Averages by Radius***" buttons.
- Upon:
    - Click ***View Top 25 Popular Manufactures*** button, jump to the **View Top 25 Popular Manufactures** task.
    - Click ***View Bathroom Statistics*** button, jump to the **View Bathroom Statistics** task.
    - Click ***View Laundry Center Report*** button, jump to the **View Laundry Center Report** task.
    - Click ***View Extra Fridge/freezer Report*** button, jump to the **View Extra Fridge/freezer Report** task.
    - Click ***View Average TV Display Size by State*** button, jump to the **View Average TV Display Size by State** task.
    - Click ***Search Manufacture / Model*** button, jump to the **Search Manufacture / Model** task.
    - Click ***View Household Averages by Radius*** button, jump to the **View Household Averages by Radius** task.

View Top 25 Popular Manufacturers

Task Decomp



Lock Types: Read-only locks on Appliance, Refrigerator, Cooker, Oven, Cooktop, Dryer, Washer, TV, Manufacturer

Number of Locks: 9 locks

Enabling Conditions:

> The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Top 25 Popular Manufacturers*** button is pressed in **View Reports.**

Frequency: They both have the same frequency.

Consistency(ACID): Critical, several queries will be needed and if ACID is not compliant, raw count and the top 25 manufacturers might not match. **View Top 25 Popular Manufacturers Each Appliance Raw Count** needs to use the result from **View Top 25 Popular Manufacturers.**

Subtasks:

> Mother task is required;
>
> Decomposition is needed;
>
> Order of subtask:
>
> > **View Top 25 Popular Manufacturers** must be run before **View Top 25 Popular Manufacturers Each Appliance Raw Count**.

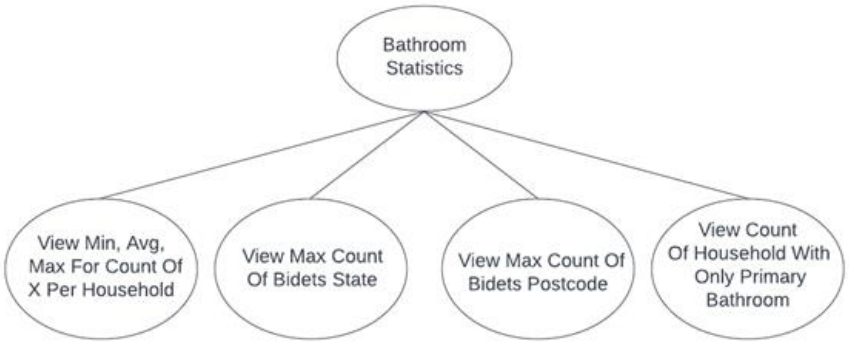
Abstract Code

- User clicked on the ***View Top 25 Popular Manufacturers*** button from the **View Reports.**
- Run **View Top 25 Popular Manufacturers** subtask:
  - Find information about Manufacturer and Appliance

        - Count: for each Manufacturer object, how many Appliance objects associated with it – how many Appliance objects are PRODUCED BY this manufacturer.
        - Sort the *ManufacturerName* values by the count as descending order, with limit of top 25 manufactures.
    - Display **Top 25 Popular Manufacturers** and each manufacturer in the list will be associated with a link for users to view detailed info about a specific manufacturer.
- Run **View Top 25 Popular Manufacturers Each Appliance Raw Count** subtask:
    - Find information about Manufacturer and Appliance in **Top 25 Popular Manufacturers list:**
        - *ManufacturerName* and *ApplianceType.*
    - For each manufacturer, count the frequency of each appliance type.
- When the link of a specific manufacturer in **Top 25 Popular Manufacturers** is clicked,
    - Display *ManufacturerName*, *ApplianceType*, and the frequency of each appliance type for the selected Manufacturer.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

View Bathroom Statistics

Task Decomp



Lock Types: Read-only lock on Bathroom, HalfBathroom, FullBathroom, Household, Location

Number of Locks: 5 locks

Enabling Conditions:

The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Bathroom Statistics*** button is pressed in **View Reports.**

Frequency: Low - they all have the same frequency

Consistency(ACID): Not critical

Subtasks:

Mother Task is not needed.

Subtasks can be done in parallel, which includes:

- **Get Min, Avg, Max of the Count of X Per Household** task
- **Max Count of Bidets State** task
- **Max Count of Bidets Postcode** task
- **Count of Household with Only Primary Bathroom** task: Query for the household with 1 Bathroom and it is primary

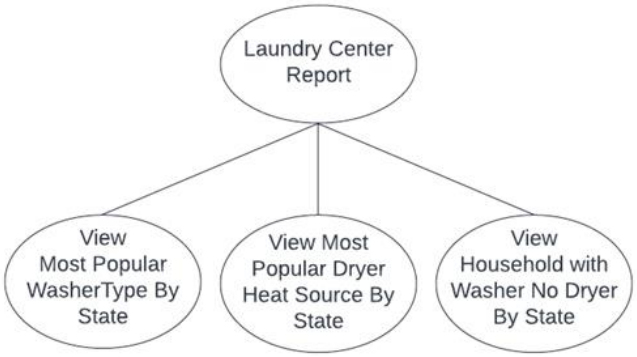
Abstract Code

- User clicked on the ***View Bathroom Statistics*** button from the **View Reports.**
- Run ***View Bathroom Statistics*** task:
  - Find information about Bathroom, HalfBathroom, FullBathroom, Household, Location.
  - Run **Get Min, Avg, Max of the Count of X Per Household** task:
    - Count the number of X(Bathroom, HalfBathroom, FullBathroom) objects associated with each Household object respectively.
    - Count the number of X(commodes, sinks, bidets, bathtubs, showers, tub/showers) per Household.
    - Calculate the Avg count of all X per household as Double data type: using number of Household objects in system and count of number of X per Household.
    - Calculate the Min and Max count of all X per household as Integer data type: order the count of X per household.
    - Display Min, Avg, Max of the count of X per Household.
  - Run **View Max Count of Bidets State** task:
    - Count the number of bidets for each State based on information about Location, Household, Bathroom.
    - Get the state with max number of bidets, then display values of this *State* and its total number of bidets.
  - Run **View Max Count of Bidets Postcode** task:
    - Based on information about Location, Household, Bathroom, Count the number of bidets for each *PostalCode* in Location.
    - Get the PostalCode with max number of bidets, then display values of this *PostalCode* and its total number of bidets.
  - Run **View Count of Household with Only Primary Bathroom** task:

    - Based on information about Location, Household, Bathroom, FullBathroom, count the number of Household objects which has only one bathroom, and this bathroom is primary.
    - Display value of count.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

View Laundry Center Report

Task Decomp



Lock Types: Read-only lock on Dryer, Washer, Appliance, Household, Location, HeatSource

Number of Locks: 6 locks

Enabling Conditions:

The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Laundry Center Report*** button is pressed in **View Reports.**

Frequency: All the subtasks have the same frequency

Consistency(ACID): Not critical

Subtasks:

Mother Task is not needed.

Subtasks can be done in parallel, including:

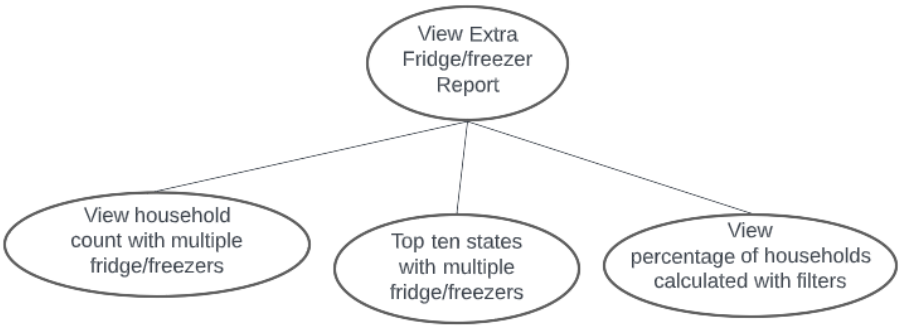
- **Most Popular Washer Type By State** task
- **Most Popular Dryer Heat Source By State** task
- **Household with Washer No Dryer By State** task

Abstract Code

- User clicked on the ***View Laundry Center Report*** button from the **View Reports**
- Run ***View Laundry Center Report*** task:
    - Find the information for Washer, Household, Location, Dryer, HeatSource.
    - Run **View Most Popular Washer Type By State** task:
        - Based on the information for Washer, Household, Location.
            - Count each *WasherType* for each *State*, and get the most common *WasherType* for each *State.*
        - Display Most Popular Washer Type By State: the output table is ordered by state ascending.
    - Run **View Most Popular Dryer Heat Source By State** task:
        - Based on the information for Dryer, Household, Location, HeatSource.
            - Count number of each HeatSource associated with Dryer for each *State*, and get the most common *HeatSourceType* for each *State.*
        - Display Most Popular Dryer HeatShource Type By State: the output table is ordered by state ascending.
    - Run **View Household with Washer No Dryer By State** task:
        - Based on the information for Dryer, Washer, Household, Location.
            - For each *State, c*ount number of Household with only Washer but no Dryer.
        - Display number of Households with Washer No Dryer By State: the output table is ordered household count descending.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

View Extra Fridge/freezer Report

Task Decomp



Lock Types: Read-only locks on Refrigerator, Appliance, Household, Location

Number of Locks: 4 locks

Enabling Conditions:

The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Extra Fridge/freezer Report*** button is pressed from **View Reports.**

Frequency: All the subtasks have the same frequency.

Consistency(ACID): not critical

Subtasks:

Mother task is needed for coordination.

Execution order of the subtasks:

- **View household count with multiple fridge/freezers** should always be run before **Top ten states with multiple fridge/freezers**
- **View percentage of households calculated with filters** should be run after **Top ten states with multiple fridge/freezers**

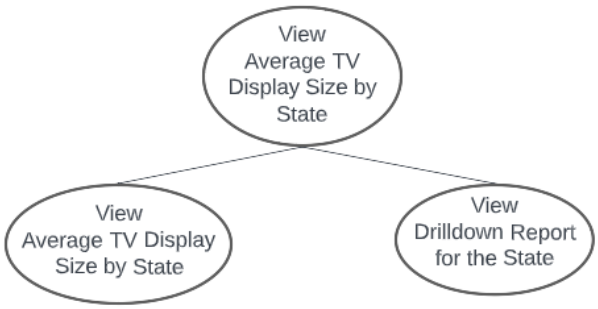
Abstract Code

- User clicked on the ***View Laundry Center Report*** button from the **View Reports**
- Run **View Extra Fridge/freezer Report task:**
  - Run **View household count with multiple fridge/freezers**
    - Find and count the number of Households that have count of Refrigerator greater than 1;
    - Display the count.
  - Run **Top ten states with multiple fridge/freezers**
    - Based on list of Households with multiple fridge/freezers, count number of Households by state. Order the list of states by descending order of the count and display.
  - For each state in the top ten list:
    - Run **View percentage of households calculated with filters**
      - Find all Households in the state
      - Based on list of Households, count number of households with:
        - multiple fridge/freezers
        - multiple fridge/freezers and chest freezer
        - multiple fridge/freezers and upright freezers
        - multiple fridge/freezers and freezers other than chest/upright freezers
      - Calculate and display the 4 percentages by dividing each count by total number of Households in the state

- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

View Average TV Display Size by State

Task Decomp



Lock Types:

Read-only lock on TV, Appliance, Household, Location

Number of Locks: 4 locks

Enabling Conditions:

The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Average TV Display Size by State*** button is pressed in **View Reports**.

Frequency:

View Average TV Display Size by State has low frequency.

View Drilldown Report for the State has a lower frequency because it is triggered by users clicking ***View Drilldown Report***

Consistency and Order(ACID): Not critical

Subtasks:

Mother task is needed for coordination.

Execution order of the subtasks:

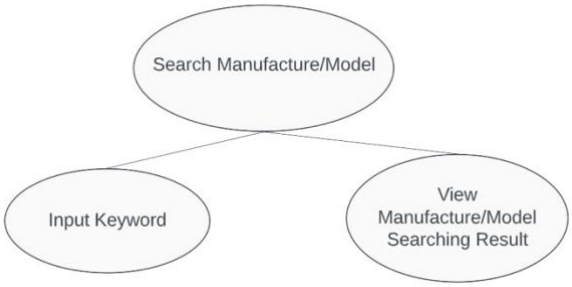
- **View Average TV Display Size by State** should always be run before **View Drilldown Report for the State**

Abstract Code

- User clicked on the ***View Household Averages by Radius*** button from the **View Reports**
- Run **View Average TV Display Size by State**
    - Find information for TV, Household, Location.
    - Count the total number of TVs in the state.
    - Calculate sum of display size of all TVs in the state.
    - Calculate the average TV display size (data type is Double) by state: dividing sum of display size by number of TVs.
    - Display Average TV Display Size by State ordered by state ascending.
- For each state, if user clicks on ***View Drilldown Report*** button associated with the state,
    - Run **View Drilldown Report for the State** subtask:
        - For this specific state, for each *DisplayType* and *MaxResolution* combination,
            - Count the number of TVs with this combination of *DisplayType* and *MaxResolution.*
            - Calculate average screen size by dividing sum of all TV screen size by number of TVs (data type is Double).
        - Display table of *DisplayType*, *MaxResolution*, and the corresponding average screen size, ordered by average screen size in descending order.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

Search Manufacture / Model

Task Decomp



Lock Types:

Lookup Manufacturer, Appliance, all are read-only

Number of Locks: 2 locks

Enabling Conditions:

> The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***Search Manufacture / Model*** button is pressed in **View Reports**.

Frequency: Same frequency for both subtasks.

Consistency(ACID): consistency is not critical.

Subtasks:

Mother task needed.

Decomposition needed. The order of subtasks:

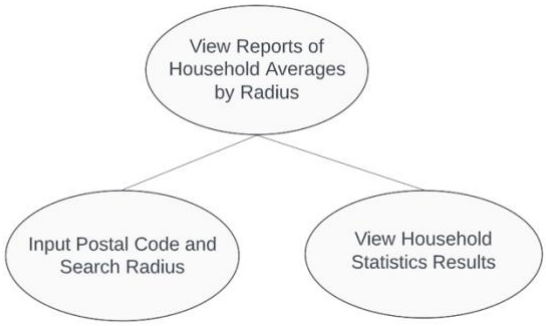
> **Input Keyword** subtask should be done before **View Manufacturer/Model Searching Result**.

Abstract Code

- User clicked on ***Search Manufacture / Model*** button, run **Search Manufacture / Model** task.
- User enters a keyword (with case Insensitivity).
- If data validation is successful for the input field, then:
    - When the ***Search*** button is clicked,
        - Get all the matched *ManufacturerName* and *ModelName.*
            - Get all the Manufacturers with part of *ManufacturerName* matches the input;
            - Get all the Appliance associated with each Manufacturer to further get all the values of *ModelName* in Appliance that related to this Manufacturer.
            - Get all the Appliance, then get all the values of *ModelName* which part of it matches the input.
        - If no match is found,
            - Let user enter it again, with an error message.
        - Else matches are found,
            - Display a distinct list(deduplication) ordered by *ManufacturerName* and *ModelName* ascending.
- Else the input field is invalid, let user enter it again, with an error message.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.

View Household Averages by Radius

Task Decomp



Lock Types: Read only locks on Location, Household, Bathroom, Appliance, HeatSource

Number of Locks: 5 locks

Enabling Conditions:

> The ***View reports/ query data*** button is pressed in **Main Menu**, and the ***View Household Averages by Radius*** button is pressed in **View Reports**.

Frequency: All subtasks have same frequency.

Consistency(ACID): consistency is not critical

Subtasks:

> Mother task is required to coordinate subtasks.
>
> Subtasks need to be done in specific order:
>
> > **Input Postal Code and Search Radius** subtask need to be done before **View Household Statistics Results** subtask.

Abstract Code

- Click on the ***View Household Averages by Radius Report*** button from the **View Reports**
- Run **Input Postal Code and Search Radius** subtask
  - Populate *PostalCode* and *SearchRadius* input Options, User enter postal code (5 digits) and select one of the radius from the dropdown list {0,5,10,25,50,100,250}.
  - If data validation is successful for the input fields *PostalCode* and *SearchRadius*, then:

        - When ***Submit*** button is clicked, run **View Household Statistics Results** subtask.
    - If any of the inputs is invalid, let user enter them again, with an error message.
- Run **View Household Statistics Results** subtask:
    - Determine the searching area based on the inputs;
    - Determine all the postal codes within this circle using Longitude and Latitude values.
    - Find information for Location, Household, Bathroom, Appliance, HeatSource.
    - Calculate Household statistics
        - Count number of Household objects in the search area; then calculate the average number of Bedroom, Bathroom, Occupant, Appliance, ratio of commodes to occupants, and the most common HeatSource type.
    - Display report results
        - Display input *PostalCode* and *SearchRadius*, the average bedroom counts(Double data type), average Bathroom counts (Double), average occupant counts (Integer), ratio of commodes to occupants (Double), average Appliance counts (Double), most common HeatSource type.
- If there is no data stored in the system corresponding to this report, an error message will be prompted to user to indicate this.