Table of Contents

1. Hemkraft Data Types:

Household

Attribute	Data type	Nullable
Email	String	Not Nullable
Square_footage	Integer	Not Nullable
Occupant	Integer	Not Nullable
Bedroom	Integer	Not Nullable
Home_type	String	Not Nullable

Postal Code

Attribute	Data type	Nullable
Postal_code	String	Not Nullable
City	String	Not Nullable
State	String	Not Nullable
Latitude	Float	Not Nullable
Longitude	Float	Not Nullable

Phone Number

Attribute	Data type	Nullable
Area_code	String	Not Nullable
Number	String	Not Nullable
Phone_type	String	Not Nullable

Appliance

Attribute	Data type	Nullable
Model_name	String	Nullable

Manufacturer

Attribute	Data type	Nullable
Name	String	Not Nullable

Bathroom

Attribute	Data type	Nullable
Sink	Integer	Not Nullable
Commode	Integer	Not Nullable
Bidet	Integer	Not Nullable
Is_primary	Boolean	Not Nullable

Half

Attribute	Data type	Nullable

Name	String	Nullable

Full

Attribute	Data type	Nullable
Bathtub	Integer	Not Nullable
Shower	Integer	Not Nullable
Tub/shower	Integer	Not Nullable

Freezer

Attribute	Data type	Nullable
Type	String	Not Nullable

Cooker

Attribute	Data type	Nullable

Oven

Attribute	Data type	Nullable
Heat_source	String	Not Nullable
Oven_type	String	Not Nullable

Cooktop

Attribute	Data type	Nullable
Heat_source	String	Not Nullable

Washer

Attribute	Data type	Nullable
Loading_type	String	Not Nullable

Dryer

Attribute	Data type	Nullable
Heat_source	String	Nullable

TV

Attribute	Data type	Nullable
Display_type	String	Not Nullable
Display_size	Float	Not Nullable
Maximum_resolution	String	Not Nullable

2. Hemkraft Constraints

Business Logical Constraints:

- **Household:**
  - The same email that has been used or recorded in the database cannot be used again
  - The email should be 320 characters long, with the domain name to be no more than 254, and local username should be no more than 64, both parts connected with a @ character
  - Square footage should be a positive integer
  - Occupants should be a positive integer
  - The number of bedrooms should be a positive integer
  - Studio should not have 0 bedroom
  - Household type should be an enum String value within {“house”, “apartment”, “townhouse”, “condominium”, “mobile home”}
  - Each household must have at least one bathroom.
- **Postal Code:**
  - The postal code that is inputted should be exact 5 digits
  - The postal code that is inputted should be within the list of the given postal codes
- **Phone Number:**
  - Phone numbers must be unique per household
  - Area codes should be exact 3 digits
  - Local numbers should be exact 7 digits
  - No dash should exist / or be stored in the database
  - Phone number type should be an enum String value within {“home”, “mobile”, “work”, “other”}
- **Bathroom:**
  - Each Bathroom is identified and associated with the household by the order it is entered into the system.
  - One Household can only have one primary Bathroom or none.
- **Half:**
  - The Half bathroom must have at least one sink, commode and/or bidet
- **Full:**
  - There may only be one primary bathroom per household
  - The Full bathroom must have at least one bathtub, shower and/or but/shower
- **Appliance:**
  - Each Appliance is identified and associated with the household by the order it is entered into the system.
  - The manufacturer's name should be an enum String value within the fixed list provided by the project.
- **Freezer:**
  - Freezer type should be an enum String value within {“Bottom freezer refrigerator”, “French door refrigerator”, “side-by-side refrigerator”, “top freezer refrigerator”, “chest freezer”, “upright freezer”}
- **Oven:**

- Oven type should be an enum String value within {“convection”, “conventional”}
- Oven must have at least one heating source: gas, electric, and/or microwave

- **Cooktop:**
  - Oven type should be an enum String value within {“gas”, “electric”, “radiant electric”, “induction”}
- **Washer:**
  - Loading type should be an enum String value within {“top”, “front”}
- **Dryer:**
  - Heat source should be an enum String value within {“gas”, “electric”} or Null
- **TV:**
  - Display type should be an enum String value within {“tube”, “DLP”, “plasma”, “LCD”, “LED”}
  - Display size should have precision 1
  - Maximum resolution should be an enum String value within {“480i”, “576i”, “720p”, “1080i”, “1080p”, “1440p”, “2160p (4k)”, “4320p (8k)”}

3. Task Decomposition with Abstract Code

3.1. Welcome / View Report

3.1.1. Task decomposition

Lock Types: None

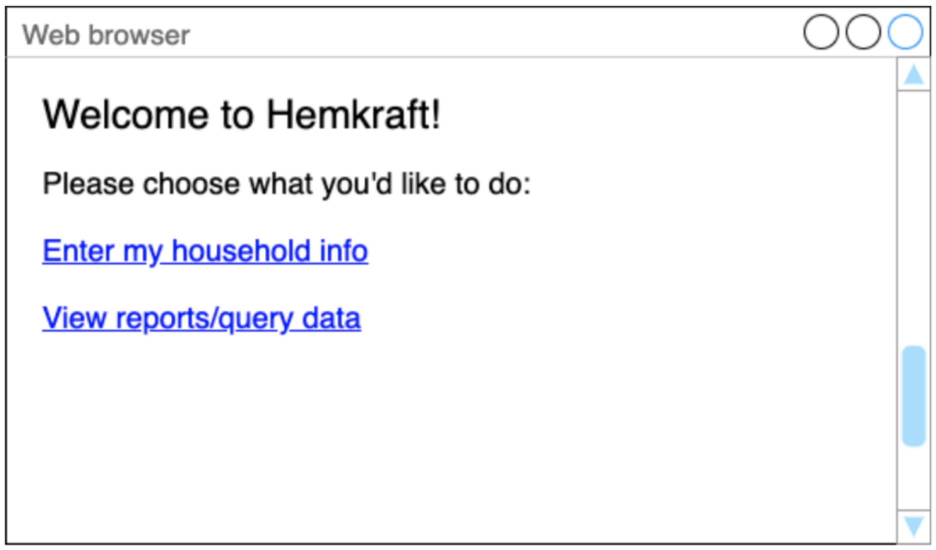
Number of Locks: None

Enabling Conditions: User access the web application

Consistency (ACID): None

Subtasks: No decomposition is needed

Welcome / View Report



Welcome to Hemkraft!

Please choose what you'd like to do:

Enter my household info

View reports/query data

3.1.2 Abstract Code:

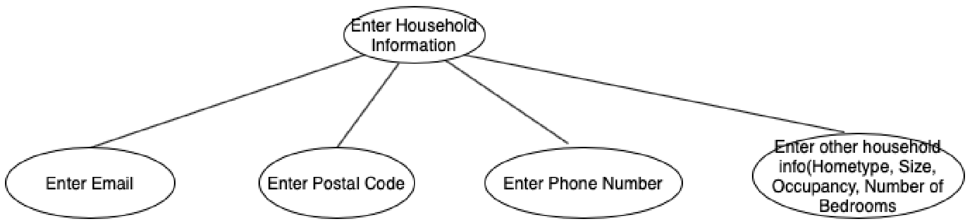
- Show **‘Enter my household info’**, ‘**View reports / query date**’ tabs.
- Do nothing until the button is pushed
- Case button pushed of:
    - Click **‘Enter my household info’,** do **‘Input household Information’** Task
    - Click ‘**View reports / query date**’, go to ‘View reports’ page
        - Do nothing until one of several buttons are pushed
        - Case button pushed of:
            - ‘Top 25 Manufacturers’, go to ‘Top 25 Manufacturers’ page
            - ‘Manufacturer/model search’, go to ‘Manufacturer/model search’ page
            - ‘Average TV display size by state’, go to ‘Average TV display size by state’ page
            - ‘Extra fridge/freezer report’, go to ‘Extra fridge/freezer report’ page

- ‘Laundry center report’, go to ‘Laundry center report’ page
- ‘Bathroom statistics’, go to ‘Bathroom statistics’ page
- ‘Household averages by radius’, go to ‘Household averages by radius’ page

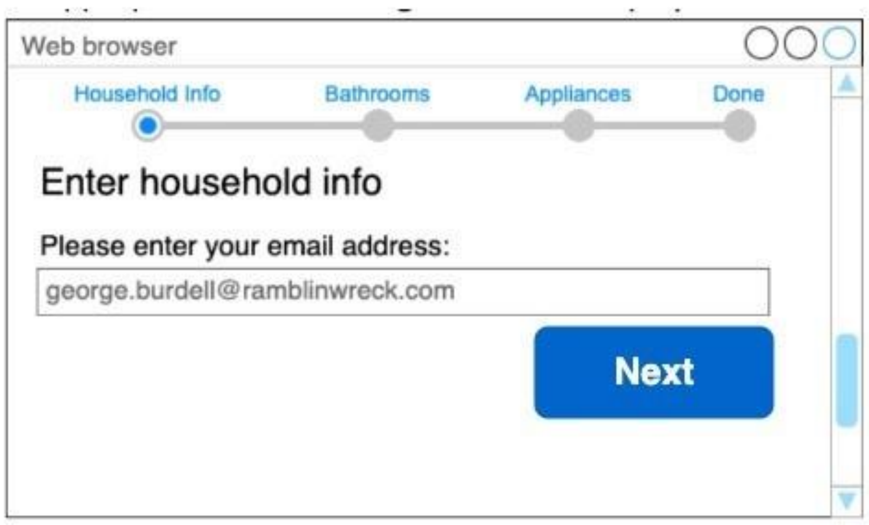
3.2. Enter Household Information

3.2.1. Task decomposition

Lock Types: Read and Write into Houshold table
Number of Locks: locks on multiple schema
Enabling Conditions: Start with landing page, and then by clicking Next in each of the entry forms.
Frequency: Assumed to be around 150 per day
Consistency (ACID): consistent is critical, order is critical.
Subtasks: Mother task is needed. Task decomposition is required. The task will be decomposed into 4 **subtasks:** Enter Email Address, Enter Postal Code, Enter Household Information)


Enter Household Information
Enter Email
Enter Postal Code
Enter Phone Number
Enter other household info(Hometype, Size, Occupancy, Number of Bedrooms

3.2.2. Abstract Code - Enter Email Address



Enter household info

Please enter your email address:

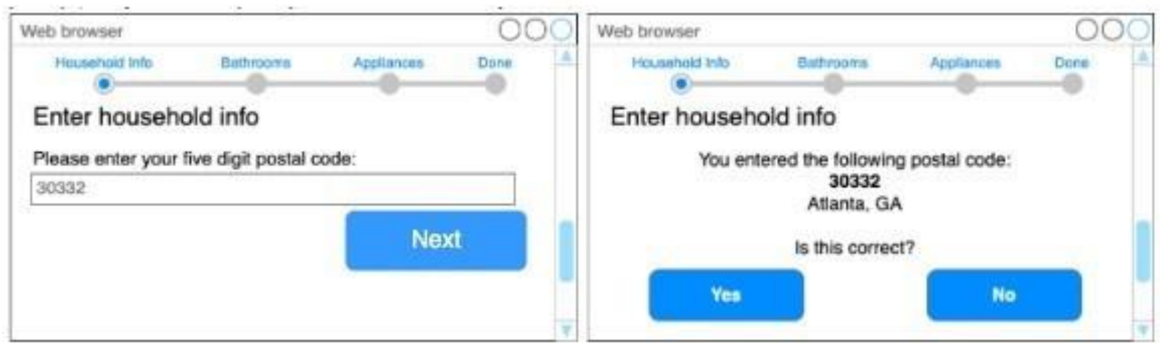
george.burdell@ramblinwreck.com

Abstract Code:

While no button is pushed, do nothing

- The user pushes the ***Next*** button, then does the following:
    - Check for email format validity: <username>@<domain>:
        - If yes: Query the database if the email address already exists:
            - If yes, display the message that reads "the email has existed and request the customer to use a different email," and empty the input field
        - Else, save the email to front-end state. then move to the next subtask, **<u>Enter Postal Code.</u>** Notice, we do not write the email into the database at the stage.
- Else: display the message that reads "this email input is invalid," and empty the input field

3.2.3. Abstract Code - Enter Postal Code



Enter household info

Please enter your five digit postal code:

30332

Enter household info

You entered the following postal code:

30332

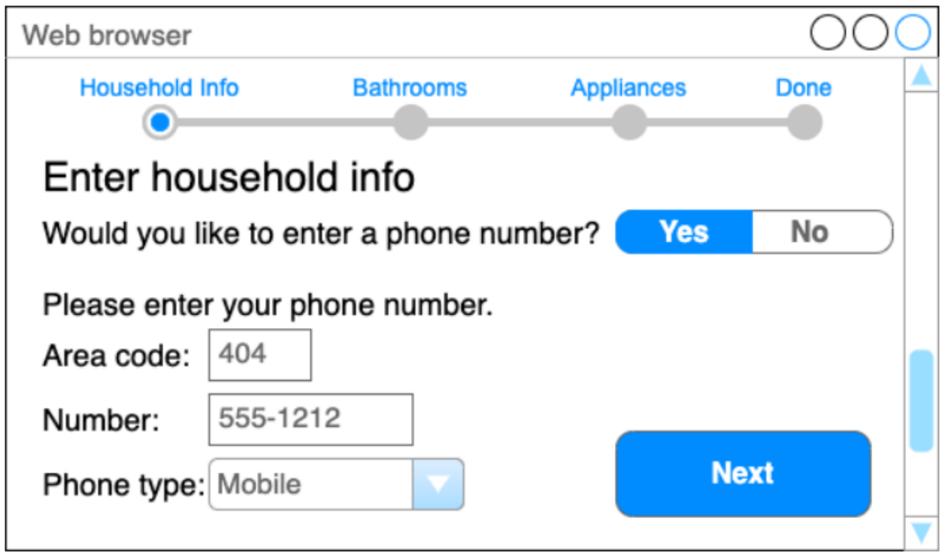
Atlanta, GA

Is this correct?

Abstract Code:

- While no button is pushed, do nothing
- When the user pushes the button ***Next***, do the following:
    - {Query the database to verify if the postal code exists:
        - If yes: display the confirmation of the postal code and the area associated with it. Ask the user to confirm:
            - If yes: Save the postal code into the front-end state, then move on to the next subtask: ***Enter Phone Number***
            - Else: restart from the beginning of this stage
        - Else: display the message that reads "invalid postal code." Clear out the form}

3.2.4. Abstract Code - Enter Phone Number



Enter household info

Would you like to enter a phone number? [✓ Yes] No

Please enter your phone number.

Area code: 404

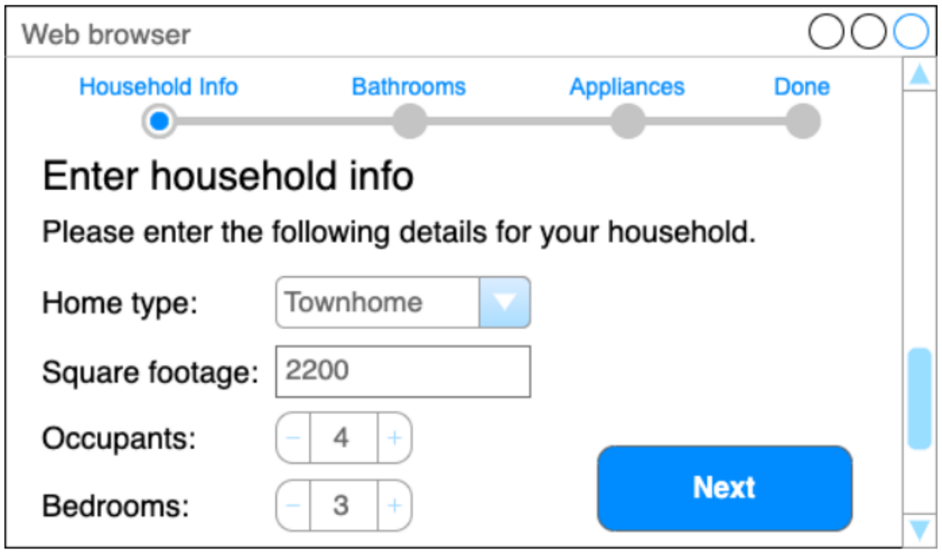
Number: 555-1212

Phone type: Mobile

Abstract Code:

- While ***no*** button for entering the phone number is pushed, do nothing
- If user choose the button ***yes***:
    - {provide the phone number form
    - The user will be required to enter the area code, and the 7-digit number, as well as choose the phone type from a drop-down form
    - When the user hits the Next button, query the database check if the phone number input has existed.
        - If yes: display the message “phone number has existed.” Reset the form
        - Else: Save the phone number into front-end state
- Else: Do nothing until the user chooses Next. Move on to the next stage, ***Enter Household Info*** (which includes Home Type, Square Footage, Occupants, Bedrooms)}

3.2.5. Abstract Code - Enter Household Information



Enter household info

Please enter the following details for your household.

Home type: Townhome

Square footage: 2200

Occupants: 4

Bedrooms: 3

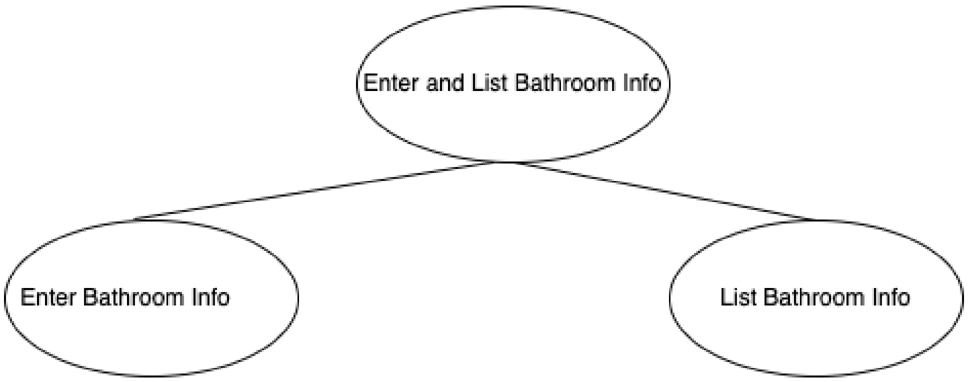
Abstract Code:

- While the button ***Next*** is not pushed, do nothing.
- When the user pushes the button, do the following:
    - {Check for integrity constraint. I.e., a studio cannot have bedrooms, the square footage is greater than 0:
        - If yes: display the message that reads “studio cannot have bedroom” or “invalid square footage”
    - Else: Save the information into front-end state, write the email into Household table, phone number into Phone Number table, and household info into the Household table. After that, move to the next task: **Enter Bathroom Info**}

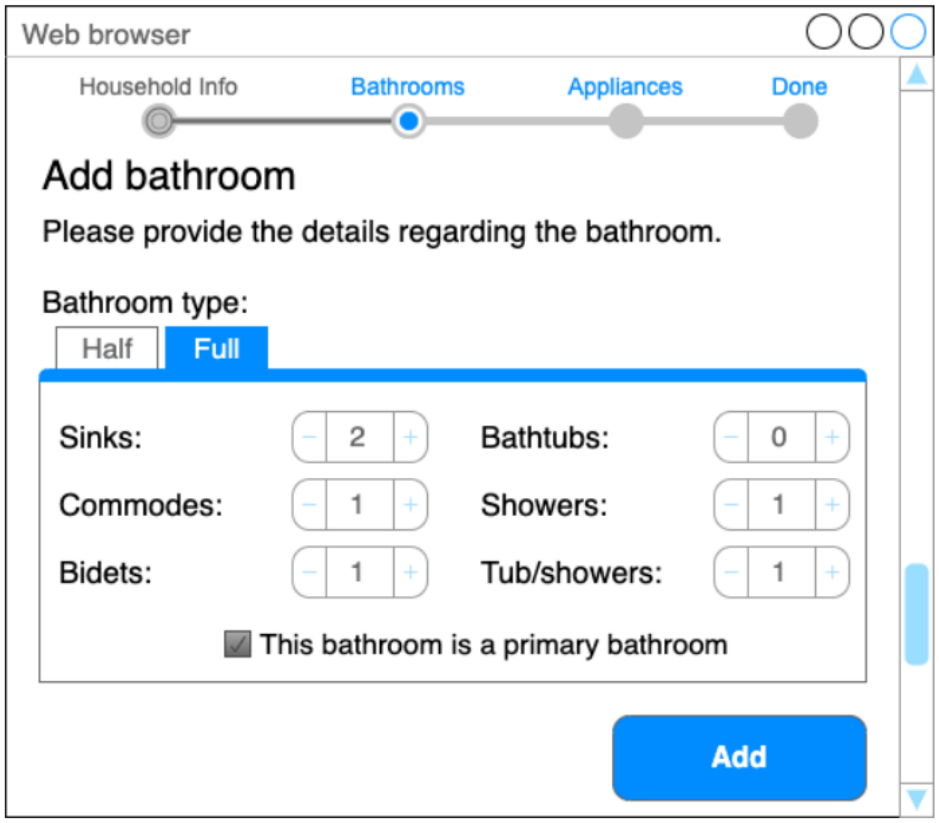
3.3. Add Bathroom Details

3.3.1. Task decomposition

Lock Types: Write-only into Bathroom, Half, and Full table
Number of Locks: Multiple locks on Bathroom, Half, and Full table
Enabling Conditions: Trigger by user choosing to proceed “Next” in the **Enter Household Form**
Frequency: Around 150 per day
Consistency (ACID): critical. Order is critical
Subtasks: **Enter Bathroom Info** and **List Bathroom** must be done subsequently with condition check required for **Enter Bathroom Info** subtask. Mother task is required to coordinate subtasks. Decomposition is needed. Order is necessary.



3.3.2. Bathroom Entry



Add bathroom

Please provide the details regarding the bathroom.

Bathroom type: Half [✓ Full]

Sinks:	2	Bathtubs:	0
Commodes:	1	Showers:	1
Bidets:	1	Tub/showers:	1

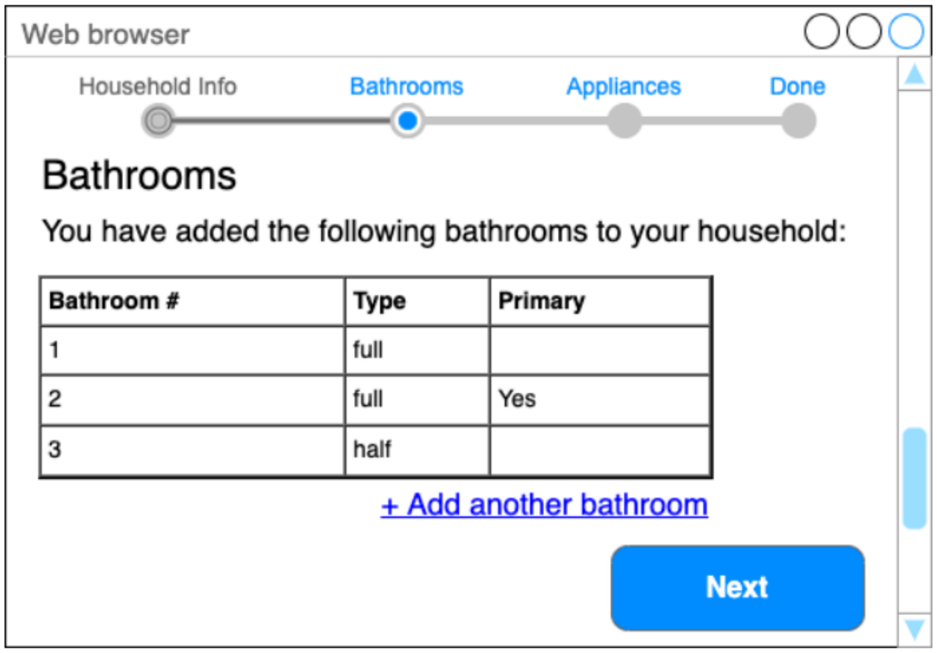
[✓] This bathroom is a primary bathroom

Abstract Code

- While the button ***Add*** is not pushed, do nothing
- If the option ***Half*** is pushed:
  - Render the **Half Bathroom** form
- If the option ***Full*** is pushed:
  - Render **Full bathroom** form
  - If the house already has a full bathroom: The option to set this bathroom as primary will be disabled
- When the ***Add*** button is pushed:

- Check the following conditions: the bathroom is a half bathroom, but it does not have at least one sink, commode, and/or bidet; or the bathroom is a full bathroom, but it does not have at least one bathtub, shower, or tub/shower.
    - Render an error message if yes.
    - Otherwise, save all the information into the front-end state, and render the following bathrooms listing}

3.3.3. List Bathroom Info



Bathrooms

You have added the following bathrooms to your household:

Bathroom #	Type	Primary
1	full	
2	full	Yes
3	half	

Abstract Code:

- {This bathroom listing shows in tabular form the bathroom spec that has been added by the previous form. The information shown is the bathroom number and their associated types and primary status
- While no button is pushed, do nothing
- If **Add Another Bathroom** is pushed: go back to the **<u>Enter Bathroom</u>** form
- If the N**ext** button is pushed:
    - write the information to the front-end state
    - Write the information into the Bathroom table, Full Table, Half Table
    - move to the next task: **Enter Appliances**}

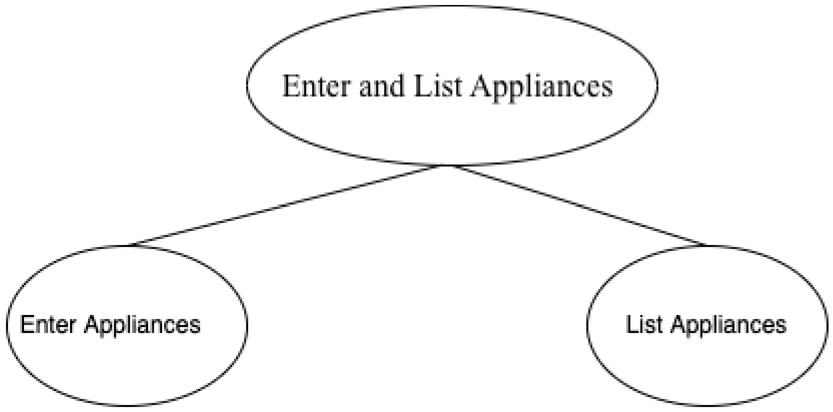
3.4. Add Appliance Details

3.4.1. Task decomposition

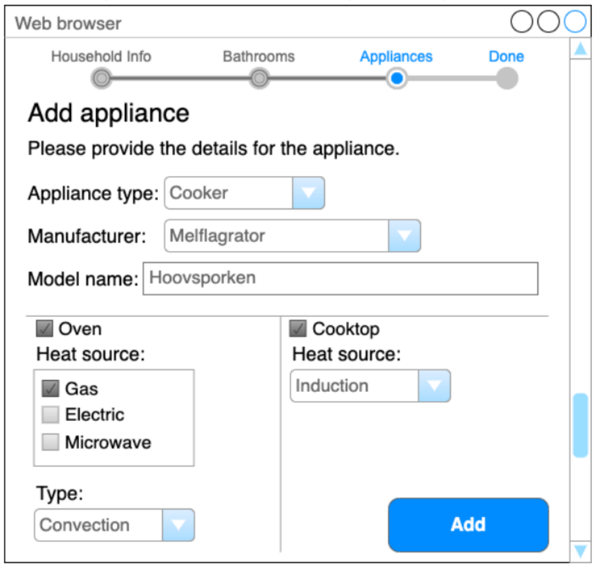
Abstract Code - Enter Appliance Info

Lock Types: Write-only to Appliance, Manufacturer, Freezer, Cooker, Oven, Cook Top, Washer, Dryer, TV
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing to proceed "Next" in the Bathroom Listing Stage

Frequency: Around 150 per day
Consistency (ACID): critical
Subtasks: Appliance entries and appliance listing must be done subsequently. In the Entry subtask, the appliance specs form is conditional on appliance type. Hence Mother task is required to coordinate subtasks. Decomposition is needed. Order is necessary.



3.4.2. Abstract Code - Enter Appliance Info



Add appliance

Please provide the details for the appliance.

Appliance type: Cooker

Manufacturer: Melflagrator

Model name: Hoovsporken

☑ Oven

Heat source:

☑ Gas

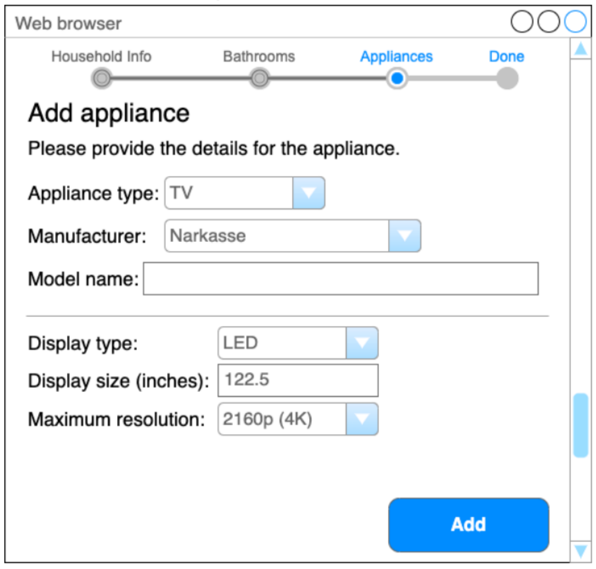
☐ Electric

☐ Microwave

Type: Convection

☑ Cooktop

Heat source: Induction



Add appliance

Please provide the details for the appliance.

Appliance type: TV

Manufacturer: Narkasse

Model name:

Display type: LED

Display size (inches): 122.5

Maximum resolution: 2160p (4K)

Abstract Code:

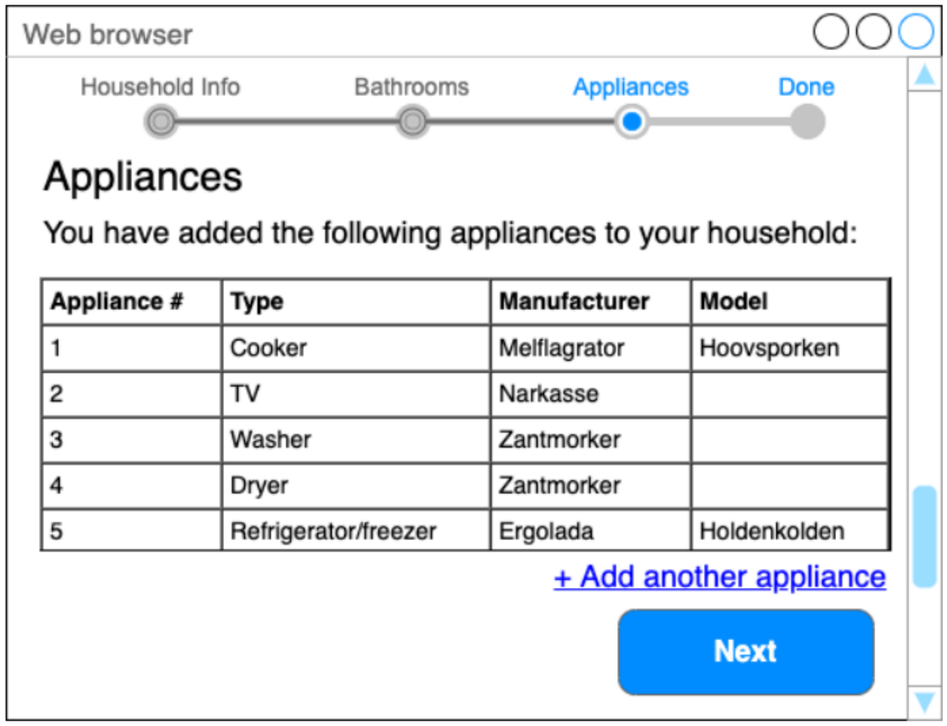
- The user chooses an appliance type:
  - {If the appliance type is a cooker:
    - Prompt the appropriate manufacturers, and model names in a dropdown menu
    - Render the cooker spec options: heat sources: gas, electric, and/or microwave, Cooktop option and its heat source: gas, electric, radiant electric, or induction.
  - If the appliance type is a TV:
    - prompt the appropriate manufacturers, and model names in a drop-down menu
    - Render the TV spec options: Display Type (tube, DLP, plasma, LCD, or LED), size input in fractional inches, the maximum resolution:480i, 576i, 720p, 1080i, 1080p, 1440p, 2160p(4K), or 4320p (8K)
  - If the appliance type is a washer:
    - prompt the appropriate manufacturers, and model names in a drop-down menu
    - Render the washer spec options: Loading type: either top or front.
  - If the appliance type is a dryer:
    - prompt the appropriate manufacturers, and model names in a drop-down menu
    - Render the dryer spec options heat source for the dryer: gas, electric, or none (For dryers which do not utilize heat, such as a condensing dryer)
  - If the appliance type is a refrigerator/freezer:
    - prompt the appropriate manufacturers, and model names in a drop-down menu
    - Render the dryer spec options heat source for the refrigerator/freezer: Bottom freezer/refrigerator, French door refrigerator, side-by-side refrigerator, top freezer refrigerator, chest freezer, upright freezer}

When the ***Add*** button is pushed: write the information into the front-end state and move to the **List Appliances** subtask

Notice 1: In this stage, each appliance will have their own specific specs. For an example, "The type of refrigerator/freezer: Bottom freezer refrigerator, French door refrigerator, side-by-side refrigerator, top freezer refrigerator, chest freezer, or upright freezer."

Notice 2: The set of manufacturers is from a finite list, which can be updated by the development team.

3.4.3. Abstract Code - Appliance Listing



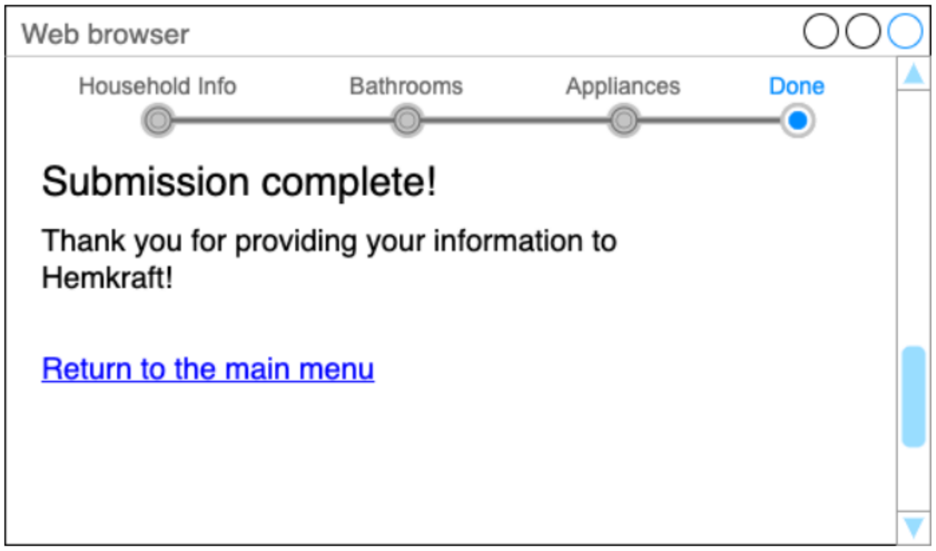
Appliances

You have added the following appliances to your household:

Appliance #	Type	Manufacturer	Model
1	Cooker	Melflagrator	Hoovsporken
2	TV	Narkasse	
3	Washer	Zantmorker	
4	Dryer	Zantmorker	
5	Refrigerator/freezer	Ergolada	Holdenkolden

Abstract Code

- This appliance listing shows in tabular form the appliances that have been added by the previous form. The information shown are the types, manufacturers, and models
- While no button is pushed, do nothing
- If "Add another appliance" is pushed: go back to the appliance adding form
- If the next button is pushed:
  - write the information to the front-end state,
  - Write the appliance info into Appliance, Manufacturer, Freezer, Cooker, Oven, Cooker Washer, Dryer, TV
  - This is the last task. Next, render the submission complete page



Submission complete!

Thank you for providing your information to Hemkraft!

Return to the main menu

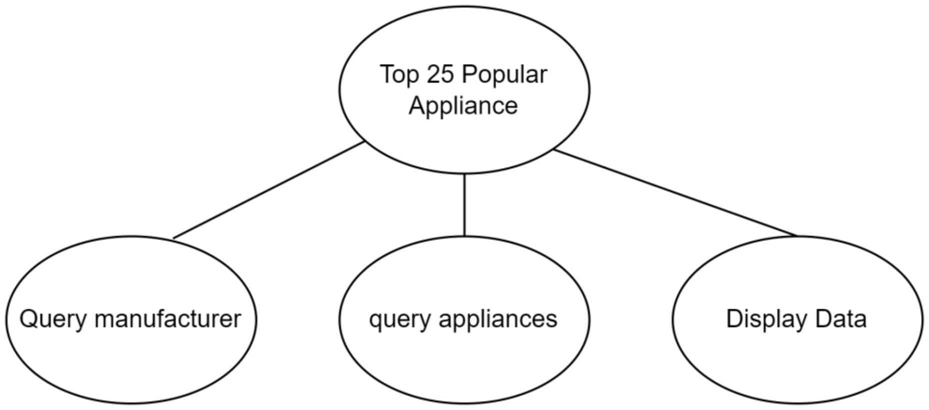
- While no button is pushed, do nothing.
- If the Back to editing information is pushed:
    - go back to the information entry form, starting from email.

3.5. Top 25 Popular Appliance Report

3.5.1. Task decomposition

Lock Types: Read-only to manufacturer, appliances, appliance type
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Top 25 Popular Appliance Report" button

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially. First, we look up the list of manufacturers. Second, we need to look up the list of appliances for each manufacturer. Decomposition is needed. Order is necessary



3.5.2. Abstract Code

Note: Big picture approach is two steps. The first step is to build dictionaries *ApplianceTypeCountDict[Manufacturer][Appliance Type]=type[1] count* and ApplianceCount*Dict[Manufacturer]=appliance count[2]*. Second step is to present(display) information.

- Initialize ApplianceCount*Dict* and *ApplianceTypeCountDict* dictionaries as sorted in descending order by value.
- Query list of *manufacturer* names
- FOREACH manufacturer's name:
    - Query list of associated *appliances*
    - Set value in dictionary [2] with key being manufacturer name and value being number of rows (appliance entries) returned.
    - FOREACH appliance in *appliances* list:
        - Query *appliance type*.
        - IF *ApplianceTypeCountDict[Manufacturer][Appliance Type] exists Increment count*
        - *ELSE add entry with value set to 1*
- Display "Top 25 Manufacturer" label
- IF ApplianceCount*Dict* has more 25 keys print first 25 entries from dictionary in two-column format: first column for keys and second column for values.

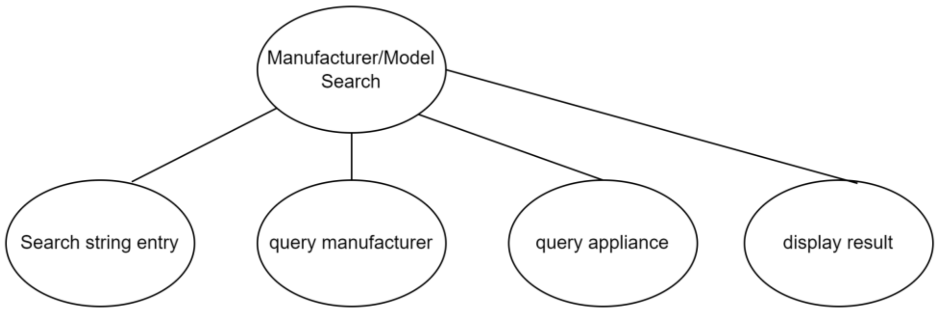
- ELSE print all entries from dictionary in two-column format: first column for keys and second column for values
- Display "Drill Down Report" label with "Pick Manufacturer" drop-down widget
- Do nothing until manufacturer is picked by user
- IF ApplianceCount*Dict[user pick]* has more 25 keys print first 25 entries from dictionary in two-column format: first column for keys and second column for values.
- ELSE print all entries from dictionary in two-column format: first column for keys and second column for values

3.6. Manufacturer / Model Search Report

3.6.1 Task decomposition

Lock Types: Read-only to manufacturer, appliance name
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Manufacturer / Model Search Report Report" button and entering search string

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially. First, we look up the list of manufacturers. Second, we need to look up list of appliance names. Decomposition is needed. Order is necessary



3.6.2. Abstract Code

Note: Big picture approach is two steps. The first step is to build dictionary *ApplianceNamesDict[Manufacturer]=name[1]*. The second step is to present(display) information.

- Do nothing until "Submit" is pressed.
- IF string entered by user in NULL display error.
- Initialize *ApplianceNamesDict*.
- Query list of *manufacturer* names
- FOREACH manufacturer's name:
    - Query list of associated *appliances and get their names*
    - IF *manufacturer* or *appliance* name contains substring entered by user

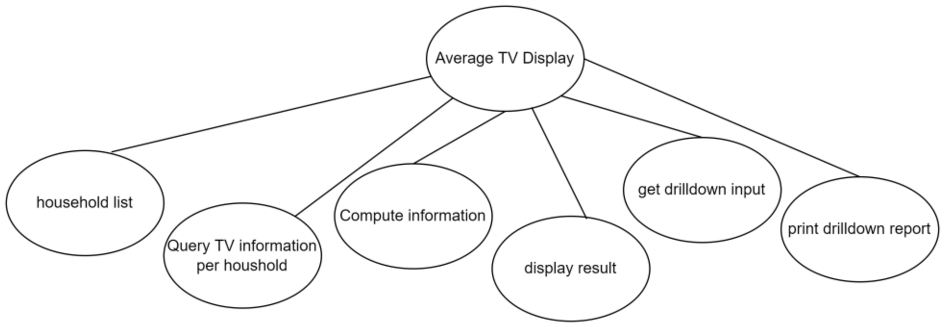
        - Set value in dictionary [1] with key being manufacturer name and value being appliance name.
- Display “Manufacturer / Model Search Report” label
- Create list of key-value pairs out of dictionary [1] sorted alphabetically.
- Display key-value pairs in two columns.
- Highlight substring entered by user in light green.

3.7. Average TV Display Size by State Report

3.7.1. Task decomposition

Lock Types: Read-only household, state code associated with it via zip code, get list of appliances of type TV, query display type, size, maximum resolution for each
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing “Average TV Display Size by State Report” button

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially.



3.7.2. Abstract Code

Note: Big picture approach is two steps. First step is to build dictionaries tvCount*Dict[State]=count[1], aggregateDisplaySizeDict[State]=aggregate[2],* householdResolutionCount*Dict[State][screenType][max_resolution]=count[3], aggregate*ResolutionSize*Dict[State][screenType][max_resolution]=aggregate[4]*. The second step is to present(display) information.

- Initialize dictionaries.
- Query list of households
- FOREACH household:
    - Query state it is associated with via zip code.
    - Query list of appliances of type TV household is associated with.
    - If household doesn’t own TV skip
    - IF State present in tvCount*Dict* increment with number of entries in list

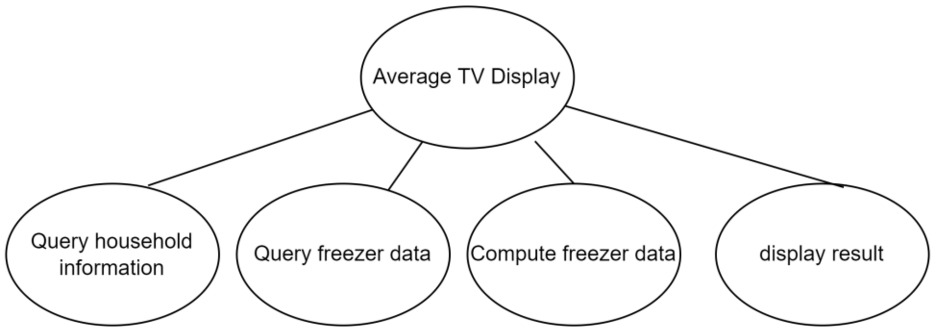
  - ELSE add entry for State in tvCount*Dict* with value set to number of entries in list
  - FOREACH appliance in TV list:
    - query *screenType, max_resolution, display_size.*
    - IF State present in *aggregateDisplaySizeDict* increment with *display_size*
    - ELSE add entry for State in *aggregateDisplaySizeDict* with value set to *display_size*
    - IF entry present in householdResolutionCount*Dict [State][screenType][max_resolution]* increment by one
    - ELSE add entry for State in householdResolutionCount*Dict [State][screenType][max_resolution]* with value set to *1*
    - IF entry present in *aggregate*ResolutionSize*Dict[State][screenType][max_resolution]* increment by display_size
    - ELSE add entry for State in *aggregate*ResolutionSize*Dict[State][screenType][max_resolution]* with value set to display_size
- Display "Average TV Display Size by State Report" label
- Create list of key-value pairs out of dictionaries [1] and [2] with key being state and value being values from dictionary [2] divided by value from dictionary [1] (aggregate sum divided by number of TVs in state). The result must be rounded to 10th digit.
- Sort list based on values in descending order.
- Display key-value pairs in two columns.
- Display "Drill Down Report" label with "Pick State" drop-down widget
- Do nothing until the state is picked by the user
- Create list of tipplets: screenType, maximum_resolution, average by iterating over dictionaries [3] and [4] and dividing aggregate sum by count.
- Round to tenths decimal.
- Sort by average in descending order.
- Display list of triplets in 3 columns.

3.8. Extra Fridge / Freezer Report

3.8.1. Task decomposition

Lock Types: Read-only household, state code associated with it via zip code, get list of appliances of type freezer, query freezer type
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Extra Fridge / Freezer Report" button

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially.



3.8.2 Abstract Code

Note: Big picture approach is two steps. The first step is to build dictionaries *householdCountDict[State]=count[1], freezerTypeDict[State][type]=count[2]*. The second step is to present(display) information.

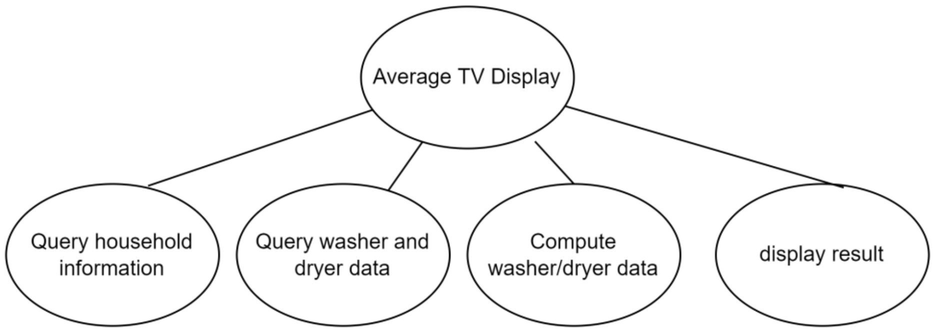
- Initialize dictionaries. *HouseholdCountDict* is ordered dictionary sorted by value in descending order.
- Query list of households
- FOREACH household:
    - Query state it is associated with via zip code.
    - Query list of appliances of type freezer household is associated with.
    - If household owns less than 2 freezers skip
    - Increment value in [1] with key set to state
    - IF list contains freezers of type *'chest'* increment [2]
    - IF list contains freezers of type *'upright'* increment [2]
    - IF list contains freezers of type *'upright'* increment [2]
    - IF list contains freezers of type different from (*'upright' or 'chest')* increment [2] with type set to 'something else'
- Display "Extra Fridge / Freezer Report" label
- Get the total household count by adding all values in [1]
- Display household count at the top of the report
- Get first 10 keys from *HouseholdCountDict* sorted dictionary.
- Using the above keys create triples by looking up in *freezerTypeDict as follows: {State, type, freezerTypeDict[State][type] divided by householdCountDict[State] multiplied by 100% and rounded to integers}*
- *Display triples in three columns*

3.9. Laundry Center Report

3.9.1. Task decomposition

Lock Types: Read-only household, state code associated with it via zip code, get list of appliances of type washer or dryer, query washer type and dryer heat source
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Laundry Center Report" button

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially.


Average TV Display
Query household information
Query washer and dryer data
Compute washer/dryer data
display result

3.9.2. Abstract Code

Note: Big picture approach is two steps. The first step is to build dictionaries *washerTypeCountDict[State][type]=count[1], dryerTypeCountDict[State][heat_source]=count[2], householdCountDict[State]=count[3]*. The second step is to present(display) information.

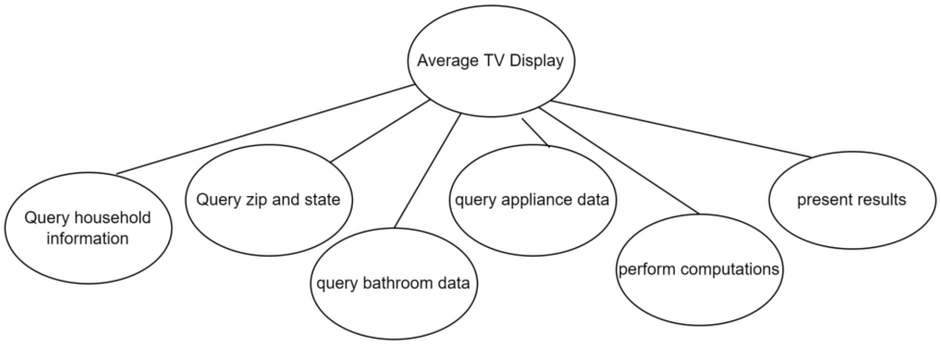
- Initialize dictionaries. *householdCountDict* is an ordered dictionary sorted by value in descending order. *WasherTypeCountDict and dryerTypeCountDict* are ordered by key in ascending order
- Query list of households
- FOREACH household:
    - Query state it is associated with via zip code.
    - Query list of appliances of type washer household is associated with and get *name* and *type*.
    - Increment inside [1] for each washer in the list or create new entry.
    - Query list of appliances of type dryer household is associated with and get *name* and *heat_source*.
    - Increment inside [2] for given state and given heat_source or create new entry.
    - If household has appliance of type washer, but does not have of type dryer increment *householdCountDict or create new entry for given state.*
- Display “Laundry Center Report” label
- Create unique list of states from from keys in dict [1] and [2].
- Sort list.
- For each value in the list:
    - lookup list of types (or heat_source types)
    - Use the first key in the list (because dictionaries are sorted)
    - Display looked up data as follows: State, washer type, heat source
- Display “Household Count” label
- Display key-value pairs from ordered dictionary [3] in 2 columns

3.10. Bathroom Statistics Report

3.10.1. Task decomposition

Lock Types: Read-only household, zip, state code associated with it via zip code, get list of bathrooms. For each bathroom query each of the attributes
Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Bathroom Statistics Report" button

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially.



3.10.2. Abstract Code

Note: First we query database and populate dictionaries mi*nDict[type], maxDict[type], aggregateDict[type], householdCount* with min/max/sum for bathroom property of each type. We also populate dictionaries bidetZipCountDict[zip] with household counts per state bidetStateCountDict[state], singleBathCount variable. The second step is displaying this information.

- Initialize dictionaries. bidetZipCountDict and singleBathCount are ordered dictionary sorted by value in descending order.
- Query list of households
- Set *householdCount variable to number of households in the list*
- FOREACH household:
    - Query *zipcode* and state associated with it via *zip*.
    - Query list of bathrooms associated with it

    - Get bathroom_count for household by adding list elements.
    - IF mi*nDict[bathrooms]*> bathroom_count:
        - mi*nDict[bathrooms]*= bathroom_count
    - IF max*Dict[bathrooms]*< bathroom_count:
        - max*Dict[bathrooms]*= bathroom_count
    - Increment *aggregateDict[bathrooms] by* bathroom_count or initialize to bathroom_count if empty

    - Get halfbathroom_count for household by adding list elements of type HALF

- IF mi*nDict[halfbathrooms]*> halfbathroom_count:
    - mi*nDict[bathrooms]*= halfbathroom_count
- IF max*Dict[halfbathrooms]*< halfbathroom_count:
    - max*Dict[halfbathrooms]*= halfbathroom_count
- Increment *aggregateDict[halfbathrooms] by half*bathroom_count or initialize to halfbathroom_count if empty

- FOREACH ‘property_name’:
    - Get propertyName_count for household by adding values of property across all bathroms in list unless it is null
    - IF minDict[‘property_name’]> property_count:
        - minDict[‘property_name’]= property_count
    - IF maxDict[‘property_name’]< property_count:
        - maxDict[‘property_name’]= property_count
    - Increment aggregateDict[‘property_name’] by property_count or initialize to property_count if empty
- FOREACH bathroom:
    - Query each attribute.
    - IF list of bathrooms for household has one entry and it has isPrimary set increment singleBathCount
    - Increment bidetZipCountDict[zip] by value of attribute *Bidet* or initialize to value of *Bidet* if zip is not present.
    - Increment bidetZipCountDict[State] by value of attribute *Bidet* or initialize to value of *Bidet* if zip is not present.

- Display “Bathroom Statistics Report” label
- Display titles of 4 columns: Type, Min, Max, Average
- Use key of dictionary minDict for first column, value of mi*nDict for second, value of* max*Dict for third and value of aggregateDict divided by householdCount for 4th column, round to 10th digit after the decimal.*
- Display “State with most bidets” label with first value in ordered bidetStateCountDict.
- Display “Zip with most bidets” label with first value in ordered bidetZipCountDict.
- Display “Households with single bath” label with first value in ordered singleBathCount.

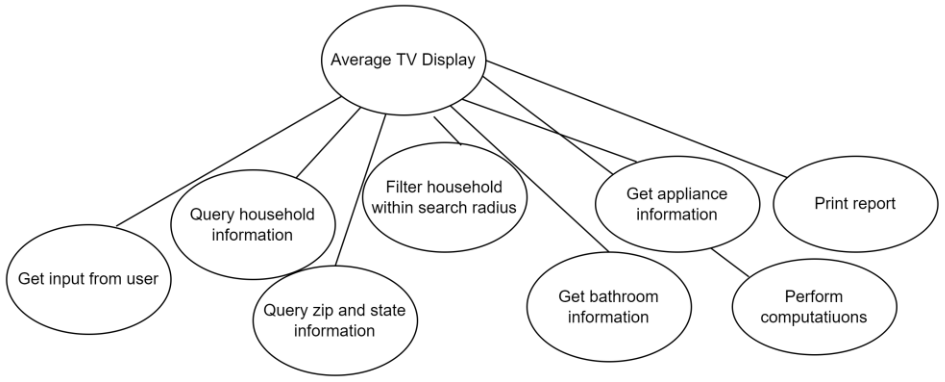
3.11. Household Average by Radius Report

3.11.1 Task decomposition

Lock Types: Read-only household, with associated zip code, get number of occupants, list of bedrooms and bathrooms with commode property associated with bedroom, lookup appliances associated with it and for dryer or cooktop appliance lookup most common heat source.

Number of Locks: several different schema constructs are needed
Enabling Conditions: Trigger by user choosing "Household Average by Radius Report" button, picking search distance and zip and pressing submit.

Frequency: Assume 50 per day or 10 per user per day
Consistency: not critical (report based on slightly outdated data is acceptable)
Subtasks: lookups must be done sequentially.


Average TV Display
Get input from user
Query household information
Query zip and state information
Filter household within search radius
Get bathroom information
Get appliance information
Perform computatiuons
Print report

3.11.2. Abstract Code

Note: Big picture approach is two steps. The first step is to update *householdInfoDict*, heatSourceDict[zip][heat_source], second step is to display information.

- Do nothing until the user hits the submit button.
- Parse input string and test against list of zipcodes provided.
- If zip is not found return error.
- Get integer value from the drop-down widget. Only valid options are possible.
- Query list of *households*.
- FOREACH household:
    - Get zip associated with household.
    - Update *householdInfoDict[zip][heat_source]=count*
    - Query database for property "distance" derived from longitude and altitude associated with zip. It is computed inside the database using harvestine formula.
    - IF distance is greater than entered by user skip.
    - Update *householdInfoDict[Email]['distance']=distance*
    - Query Occupant property
    - Update *householdInfoDict[Email]['occupant']=Occupant*
    - Query Bedroom property
    - Update *householdInfoDict[Email]['Bedroom']=Occupant*
    - Query list of bathrooms
    - Update *householdInfoDict[Email]['bathCount']=bathroom count*
    - FOREACH bathroom:
        - Query *Commode* property
        - *householdInfoDict[Email]['Commode']=Commode*
    - Query *Appliances*
    - *householdInfoDict[Email]['applianceCount']=appliance count*

    - *FOREACH appliance of type dryer or Cooker:*
        - Query *heat_source*
        - Increment *heatSourceDict[zip][heat_source]=count or initialize to one*
- *Display per-household information by iterating over householdInfoDict.*
- *Get list of unique zipcodes by iterating over householdInfoDict.*
- *FOREACH zip:*
    - *Get average bathroom count by iterating over householdInfoDict, accumulating bathroom counts per household. Then divide by number of households and perform appropriate rounding to tenths decimal.*
    - *Get average occupant count by iterating over householdInfoDict, accumulating occupant counts per household. Then divide by number of households and perform appropriate rounding to integer.*
    - *Get ratio of commodes to occupants count by iterating over householdInfoDict, accumulating occupant counts per household and do same for commodes. Then divide number of commodes by number of occupants rounded to nearest tenths*
    - *Get average appliance count by iterating over householdInfoDict, accumulating appliance counts per household. Then divide by number of households and perform appropriate rounding to tenths digit.*
    - *Find the most common heat_source type for zipcode by iterating over keys of heatSourceDict[zip] and finding heat_source key with greatest count. If no keys are present use empty string.*
- *Present above information in 5 columns: zip, distance, average occupant, commodes-per-occupant, average appliance, common heat_source*