## Conventions:

**BoldUnderline**: Form

***Bold Italics*** : Button/link names

**Bold**: Task

*Italics*: Form Input Fields

*Italics*: Key for the data base

## ​​Main Menu

### Task Decomposition

* Enabling condition: Triggered after accessing the web
* Frequency: Same frequency
* Database interaction: No read/write/update/delete/insert from database
* Lock Types: No lock
* Sequence of subtask: No subtask, no decomposition is needed
* Consistency: Not critical given there is no update of data

### Abstract Code

* Show “***Enter my household info***” and *“****View Reports/Query Data****”* buttons.
* If “***Enter my household info***” button is pushed, perform **Enter Household Info** task
* If “***View Reports/Query Data***” button is pushed, perform **View Report** task

## Enter Household Info

### Task Decomposition

* Enabling condition: Triggered after user pressed the “*Enter my household info*” button
* Frequency: Same frequency
* Database interaction: Read and Insert from HOUSEHOLD table, Read from LOCATION table
* Lock Types: 2 read locks on HOUSEHOLD and LOCATION, 1 write lock on HOUSEHOLD
* Sequence of subtask: Mother task is needed. Order is necessary
  + Validate User Inputs -> Pop Out Error Message Window or Save Data To Household Table -> go back to Add Household or go to Add Appliance
* Consistency: not critical, assume there wont be another user entering info using the same address

### Abstract Code

* If “***Next***” button is pushed,
  + **Validate User Inputs** as below:
    - If the *email* input already exists in HOUSEHOLD table:
      * Error message includes “Email already exists!”
    - If the *postal code* doesnt exist in LOCATION table:
      * Error message includes “Postal code is not correct!”
    - If data validation of *Home Type, Square footage* (whole number) doesnt pass:
      * Error message includes “Please enter valid Home Type/Square footage”
    - If “No Heat” box is not ticked, and data validation of *Themostat setting for heating* (as a whole degrees Fahrenheit) doesnt pass:
      * Error message includes “Please enter valid themostat setting for heating”
    - If “No Cooling” box is not ticked, and data validation of *Themostat setting for cooling* (as a whole degrees Fahrenheit) doesnt pass:
      * Error message includes “Please enter valid themostat setting for cooling”
  + If there is error message, **Pop Out Error Message Window**  and **Go Back to Enter Household**
  + If there is no error, **Save Data To Household Table** and **Go To Add Appliance**

## Add Appliance

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Next***” button in Enter Household Info form
* Frequency: Different fields’ frequency can be different, dependent on which Appliance/Method user selects
* Database interaction: Read from HOUSEHOLD table, Multiple times of Insert into APPLIANCE and MANUFACURER table
* Lock Type: 1 read lock for HOUSEHOLD table, 2 write lock for APPLIANCE and MANUFACURER table
* Sequence of subtask: Mother task is needed since there is order of sequence in subtask, **Display Appliance Fields Based on User Selection** -> **Run Data Validation of the Entered Fields -> (Pop Out Error Message Window and Go Back to Add Appliance)** or (**Create new Appliance to APPLIANCE database** and and **go to Appliance Listing)**
* Consistency: critical, data should be persisted to the database when saving from the screen

### Abstract Code

* **Display Appliance Fields Based on User Selection**
  + If user selects *Air Handler* 
    - Displays fields of *manufactuer, model name, BTU and Fan RPMs*
    - Displays heating/cooling method
      * If user selects *air conditioner,* displays *EER*
      * If user selects *heater*, displays *energy source*
      * If user selects *heat pup*, displays *SEER and HSPF*
  + If user selects *Water Heater*
    - Displays fields of manufactuer, model name, BTU, tank size, current temperature setting and energy source
* If “***Add***” button is pushed,
  + **Run Data Validation of the Entered Fields** 
    - If *BTU* doesnt esit or not in whole number, add to error message
    - If *Manufactuer or Model Name* is Null, add to error message
    - If *Air Handler* is selected
      * if *RPM* is Null or not in whole number, add to error message
      * If *air conditioner* is selected , *EER* is Null or not in decimal number to the tenth decimal point, add to error message
      * if *heater* is selected and *energy source* is Null, add to error message
      * if *heat pup* is selected, *SEER or HSPF is Null* or not in decimal number to the tenth decimal point, add to error message
    - If *Water heater* is selected,
      * if t*ank size* is Null *or* not in decimal number to the tenth decimal point , add to error message
      * if *current temperature setting* is not Null but not in whole number, add to error message
      * If *energy source* is Null, add to error message
    - If there is error message, **Pop Out Error Message Window** and **Go Back to Add Appliance**
    - If there is no error, based on user inputs, **Create new Appliance to APPLIANCE database** and **go to Appliance Listing**

## Appliance Listing

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Add***” button in Add Appliance form
* Frequency: The form has relatively high frequency, appear each time after a new appliance is added; Each field has the same frequency
* Database interaction: Read/Delete from APPLIANCE table
* Lock Type: 1 read and 1 write lock for APPLIANCE table
* Sequence of subtask: Mother task is neede since there is subtask sequence, **View Appliance** -> **1) Delete or 2) Add Appliance** -> **1) Save Data and View Appliance again or 2) go to Add Power Generation**
* Consistency: critical, data should be persisted to the database when saving from the screen

### Abstract Code

* **View Appliance** 
  + Query the Appliance table and display a list of appliances with info of Type, Manufacturer and Model
* If **“*delete*”** button is pushed, the related record is to**Delete Appliance** from Appliance table
* If **“*Add another appliance*”** button is pushed, go to **Add Appliance**
* If **“*Next*”** button is pushed, go to **Add Power Generation**

## Add Power Generation

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Next***” button in Appliance Listing form
* Database interaction: Read from HOUSEHOLD table, Insert into POWERGENERATION table
* Lock Type: 1 read lock for HOUSEHOLD table, and 1 write lock for POWERGENERATION table
* Sequence of subtask: Mother task is not needed. Decomposition is not needed
* Consistency: not critical

### Abstract Code

* If user presses ***Skip*** button
  + If public utility in HOUSEHOLD table is empty, then go to Power Generation Listing form
  + Else: pop up Error Window “Please enter power generation information”
* If user presses ***Add***  button
  + If *generation type* is not selected, add to error message
  + If *average monthly kilowatt hours* is Null or not in whole number, add to error message
  + If *battery storage capacity* is not Null but not in whole number, add to error message
  + If there is error message, Pop Out Error Message Window and Go Back to Add Power Generation
  + If there is no error, based on user inputs, Save Data to POWERGENERATION database and go to Power Generation Listing

## Power Generation Listing

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Add***” button in Add Power Generation form
* Database interaction: Read/Delete from POWERGENERATION table, Read from HOUSEHOLD table
* Lock Type: 1 read and 1 write lock for POWERGENERATION table, 1 read lock for HOUSEHOLD table
* Sequence of subtask: Mother task is needed since there is subtask sequence, **View Power Generation** -> **1) Delete or 2) Add PowerGeneration** -> **1) ( Save Data and View PowerGeneration again) or 2) (go to Add Power Generation task)**
* Consistency: critical, data should be persisted to the database when saving from the screen

### Abstract Code

* **View Power Generation** 
  + Query the POWERGENERATION table and display a list of Power Generation with info of Type, Monthly Kwh and MoBattery kWh
* If **“*delete*”** button is pushed, the related record is to**Delete Power Generation** from POWERGENERATION table
* If **“*Add more power*”** button is pushed, go to Add Power Generation form
* If **“*Finish*”** button is pushed,
  + If there is no power added and the public utility is not empty in HOUSEHOLD table
    - Pop up Window to remind adding power, then go back to Add Power Generation form
  + else:
    - Go to Wrapping Up form

## Wrapping Up

### Task Decomposition

* Enabling condition:Triggered after user pressed the “***Finish***” button in Power Generation Listing form
* Database interaction: No read/write/update/delete/insert from database
* Lock Types: No lock
* Sequence of subtask: No subtask, no decomposition is needed
* Consistency: Not critical given there is no update of data

### Abstract Code

* Show “***Return to the main menu”*** button.
* If “***Return to the main menu***” button is pushed, perform **Main Menu** task

## View reports

### Task Decomposition

* Enabling condition:Triggered after user pressed the “***View Reports/Query Data***”button in the Main Menu page
* Database interaction: No read/write/update/delete/insert from database
* Lock Types: No lock
* Sequence of subtask: No subtask, no decomposition is needed
* Consistency: Not critical given there is no update of data

### Abstract Code

* Show links for the following sections: ***Top 25 popular manufacturers, Manufacturer/model search, Heating/cooling method details, Water heater statistics by state, Off-the-grid household dashboard, Household averages by radius***
* If each of the link is pushed by the user, perform the corresponding task with the same name
  + *e.g.*, if “***Top 25 popular manufacturers****”* is pushed, perform **Top 25 popular manufacturers**
* If “***Finish***” button is pushed, perform **Main Menu** task

## Top 25 popular manufacturers

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Top 25 popular manufacturers***” button in View Reports form
* Database interaction: Read from APPLIANCE and MANUFACTURER table
* Lock Types: 1 read lock from APPLIANCE and 1 read lock from MANUFACTURER table
* Consistency: Not critical given there is no update of data
* Sequence of subtask: The process is **list the top twenty-five manufacturers 🡪 drilldown report** for any certain manufacturer (required by users, can be multiple times). All tasks must be done based on the order conducted by the user. Mother task is required to coordinate subtasks. since the user has the option to look at the drilldown report for any certain manufacturer.

### Abstract Code

* **List the top twenty-five manufacturers**
  + Query the APPLIANCE table and MANUFACTURER table (with APPLICANCE left joined by MANUFACTURER via common key manufacturer\_name)
  + Display the list of top 25 manufacturers with the most appliances, with columns of manufacturer name, raw count presented
  + At each row show a button ***drilldown report*** to enable the subtask **drilldown report**
  + If ***drilldown report*** pushed by the user, perform **drilldown report** for the specific manufacturer in that row
* **Drilldown report**
  + Query the APPLIANCE table and MANUFACTURER table
  + Display manufacturer name at the top
  + Display a table, with columns of appliance type, and appliances count presented
* If “***Finish***” button is pushed, perform **View Reports** task

### Manufacturer/model search

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Manufacturer/model search***” button View Reports form
* Database interaction: Read from MODEL and MANUFACTURER table
* Lock Types: 1 read lock from MODEL and 1 read lock from MANUFACTURER table
* Consistency: Not critical given there is no update of data
* Sequence of subtask: No subtask, no decomposition is needed

### Abstract Code

* User enters a string of *keyword* in the input field
* Query the MODEL table and MANUFACTURER table (with MODEL left joined by MANUFACTURER via common key manufacturer\_name), search for the entries where *keyword* is either a substring of the attribute model\_name or manufacturer\_name
* Display result with column of model\_name and manufacturer\_name, ordered by manufacturer name ascending and model name ascending
* Highlight the any substring containing *keyword* in both columns (if any) of model\_name and manufacturer\_name
* If “***Finish***” button is pushed, perform **View Reports** task

### Heating/cooling method details

### Task Decomposition

* Enabling condition:Triggered after user pressed the “***Heating/cooling method details***” button in View Reports form
* Database interaction: Read from HOUSEHOLD, APPLIANCE, (AIR HANDLERS, AIR CONDITIONER, HEATER, HEAT PUMP)
* Lock Types: Read-only lock for each of the table above
* Sequence of subtask: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary. Below is the list of subtasks:
  + **Air Conditioner Summary**
  + **Heater Summary**
  + **Heat bump Summary**
* Consistency: Not critical given there is no update of data

### Abstract Code

* Query the HOUSEHOLD table and APPLIANCE table (with APPLIANCE left joined by HOUSEHOLD via common key email)
* group by the same household type, display household\_type
* **Air Conditioner Summary**
  + for each household type, find all the appliances that belong to air conditioner, calculate sum of rows as count\_ac, average(BTU) as average\_BTU\_ac, average(RPM) as average\_RPM\_ac, average(EER) as average\_EER\_ac
  + for each household type, display the corresponding count\_ac, average\_BTU\_ac, average\_RPM\_ac, average\_EER\_ac
* **Heater Summary**
  + for each household type, find all the appliances that belong to heater, calculate sum of rows as count\_ht, average(BTU) as average\_BTU\_ht, average(RPM) as average\_RPM\_ht; argmax(count\_values(energy source)) as common\_energy\_source\_ht
  + for each household type, display the corresponding count\_ht, average\_BTU\_ht, average\_RPM\_ht, common\_energy\_source\_ht
* **Heat bump Summary**
  + for each household type, find all the appliances that belong to heat bump, calculate sum of rows as count\_hb, average(BTU) as average\_BTU\_hb, average(RPM) as average\_RPM\_hb, average(SEER) as average\_SEER\_hb, average(HSPF) as average\_HSPF\_hb
  + for each household type, display the corresponding count\_hb, average\_BTU\_hb, average\_RPM\_hb, average\_SEER\_hb, average\_HSPF\_hb
* If “***Finish***” button is pushed, perform **View Reports** task

### Water heater statistics by state

### Task Decomposition

* Enabling condition: Triggered after user pressed the “***Water heater statistics by state***” button in View Reports form
* Database interaction: Read from WATERHEATER table, Read from HOUSEHOLD table, Read from LOCATION table
* Lock Types: 1 read lock for WATERHEATER table, 1 read lock for HOUSEHOLD table, 1 read lock for LOCATION table
* Sequence of subtask: The process is ***List water heater statistics by state* 🡪 Drilldown report** for any state (required by users, can be multiple times). All tasks must be done based on the order conducted by the user. Mother task is required to coordinate subtasks. since the user has the option to look at the drilldown report for any state.
* Consistency:Not critical given there is no update of data

### Abstract Code

* ***List water heater statistics by state***
  + Query the WATERHEATER table,the HOUSEHOLD table and LOCATION table (with HOUSEHOLD left joined by LOCATION via common key postal code. Then left join this table with WATERHEATER table with household email)
  + Display a table with the state’s abbreviation, the average water heater tank size, the average water heater BTUs, the average water heater temperature setting), the count of water heaters where a temperature setting has been provided, and the count of water heaters where no

temperature setting has been provided, sorted by state abbreviation ascending.

* + At each row show a button ***drilldown report*** to enable the subtask **drilldown report**
  + If ***drilldown report*** pushed by the user, perform **drilldown report** for the specific state in that row
* **Drilldown report**
  + Filter the main table to household located in selected state
  + Display state name at the top
  + Display a table grouped by energy source, the minimum water heater tank size , the average

water heater tank size, the minimum temperature setting, the average temperature setting, the maximum temperature setting. Energy sources are ordered in ascending order.

* If “***Finish***” button is pushed, perform **View Reports** task

### Off-the-grid household dashboard

### Task Decomposition

* Enabling condition: Triggered after user pressed the “ ***Off-the-grid household dashboard***” button in View Reports form
* Database interaction: Read from HOUSEHOLD table, read from LOCATION table, Read from WATERHEATER table, Read from APPLIANCE table, Read from POWERGENERATOR table
* Lock Types: 1 read lock for HOUSEHOLD table, 1 read lock for LOCATION table, 1 read lock for WATERHEATER table, 1 read lock for APPLIANCE table, 1 read lock for POWERGENERATOR table
* Sequence of subtask: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary. Below is the list of subtasks:
  + **State with most off-grid**
  + **Average battery storage capacity per battery**
  + **Percentage of power generation type**
  + **Percentage of household type**
  + **Average water heater tank size**
  + **Min, max, average BTU**
* Consistency: Not critical given there is no update of data

### Abstract Code

* If user clicks on any of the links, jump to the corresponding report form

### Household averages by radius

### Task Decomposition

* Enabling condition:Triggered after user pressed the “ ***Household averages by radius***” button in View Reports form
* Database interaction: Read from HOUSEHOLD table, Read from LOCATION table
* Lock Types: 1 read lock for HOUSEHOLD table, 1 read lock for LOCATION table
* Sequence of subtask: No subtask, no decomposition is needed
* Consistency: Not critical given there is no update of data

### Abstract Code

* User enters string of *postal code* and *radius* in the input field
* If postal code enter is invalid, display error message “invalid postal code”
* Query the HOUSEHOLD table and LOCATION table (with HOUSEHOLD left joined by LOCATION via common key postal code), calculate distance between household and *postal code*
* Filter to only keep household within input *radius*
* Display a table with the *postal code*, the search *radius*, the count of households, the count of households for each household type, the average square footage, the average heating temperature, the average cooling temperature, which public utilities are used, the count of “off-the-grid” homes, the count of homes with power generation, the most common generation method for all households with power generation, the average monthly power generation per household and the count of households with battery storage
* If “***Finish***” button is pushed, perform **View Reports** task