Solar/Roofing Drawing API – Feedback:

Solar:

1. On step 1: Please add zoom in/out buttons (a plus and minus icon) and “reset” link or button. The buttons will zoom the size of the house in or out.. and the “reset button will allow the user to move the house around again to set it. The “Next” button can be renamed to “Set” and go to step2, and where “set” currently is now (upper right) can be the new reset button.

Zoom in/out buttons added with specified functionality.

Next button renamed to “Set”

“Set” button renamed to “Reset” for steps 1 & 2 with same functionality.

1. On step 2: Add a “Restart” button to clear everything and go back to the beginning step.

Also the “zoom” should be locked, so the user is unable to zoom or modify the satellite image at this point.

Map zoom completely locked in step 2 & 3. Do you also need to lock map scroll?

Restart button not added because I was not able to differentiate functionality of “Clear” & “Restart” button. As “Clear” button is already there with same functionality you specified for “Restart”.

**Following points need inventory (panel data etc.) database to complete.**

1. The panel sizes need to be sized properly, but this probably depends on the dimensions from the database table. I will enter in the dimensions in a new table today and get that information for you to connect to this tool.
2. The mounting plane sizes are not correct yet.. (needs to be panel size (250, 280 watt, etc.) multiplied by the number of panels. I’ll provide the panel size (dimensions and wattage) in a table so this can be calculated based off panel size.
3. Also our panels should be black silicon (not blue). I will send a picture of what they look like.
4. Shading and tilt: Shading will pull from the “shading” table in the database. The selectbox will use the shadingName value from the table, and the shadingFactor as the value.   
     
      
     
   These will change periodically so needs to be pulled dynamically. The “Tilt” will be entered in by the user. All of this information needs to be sent via API to PVwatts to obtain the mounting plane’s output in kilowatts and stored in the mountingPlanes table.  
     
   Here’s what the API code looks like inside the application:

/\* dmxDataSet name "PVwattsAPI" \*/

jQuery.dmxDataSet(

{"id": "PVwattsAPI", "url": "https://developer.nrel.gov/api/pvwatts/v5.json?api\_key={{PVwattsAPIkeys.data[0].pvWattsAPIkey}}&address={{((solarEntryDetails.data[0].siteAddress ==NULL)||(solarEntryDetails.data[0].siteAddress =='')).then( solarEntryDetails.data[0].address, solarEntryDetails.data[0].siteAddress )}}+{{((solarEntryDetails.data[0].siteCity ==NULL)||(solarEntryDetails.data[0].siteCity =='')).then( solarEntryDetails.data[0].city, solarEntryDetails.data[0].siteCity )}}+{{((solarEntryDetails.data[0].siteState ==NULL)||(solarEntryDetails.data[0].siteState =='')).then( solarEntryDetails.data[0].state, solarEntryDetails.data[0].siteState )}}+{{((solarEntryDetails.data[0].siteZip ==NULL)||(solarEntryDetails.data[0].siteZip =='')).then( solarEntryDetails.data[0].zip, solarEntryDetails.data[0].siteZip )}}&system\_capacity={{$FORM.nrelSysCapacityKW}}&azimuth={{$FORM.mountPlaneAzimuth}}&tilt={{$FORM.mountPlaneTilt}}&array\_type=1&module\_type=0&losses={{$FORM.mountPlaneShadingFactor}}", "dataType": "jsonp"}

);

/\* END dmxDataSet name "PVwattsAPI" \*/

* api\_key={{PVwattsAPIkeys.data[0].pvWattsAPIkey}} is a random key that is pulled from a database view.
* The address information is pulled through, but since we now have latitude and longitude coordinates, those need to save to the database and need to replace all of the address information currently in the code above.
* system\_capacity={{$FORM.nrelSysCapacityKW}} will be the size of the Mounting plane (number of panels X panel wattage / 1000 = KW)
* azimuth={{$FORM.mountPlaneAzimuth}} will be the azimuth pulled from the satellite image
* losses={{$FORM.mountPlaneShadingFactor}} will be the Shading Factor pulled from the table and selected in the selectbox.

This is the dmxZone code used to add values to the mountingPlanes table, with information about the mounting plane as well as information returned from the PVwatts API. (annual and monthly outputs)

/\* dmxDatabaseAction name "mountingPlaneADD" \*/

jQuery.dmxDatabaseAction(

{"id": "mountingPlaneADD", "url": "dmxDatabaseActions/mountingPlaneADD.php", "data": {"solarEntryID": "{{$URL.solarEntryID}}", "mountPlaneShadingFactor": "{{$FORM.mountPlaneShadingFactor}}", "mountPlaneACJAN": "{{$FORM.mountPlaneACJAN}}", "mountPlaneACFEB": "{{$FORM.mountPlaneACFEB}}", "mountPlaneACMAR": "{{$FORM.mountPlaneACMAR}}", "mountPlaneACAPR": "{{$FORM.mountPlaneACAPR}}", "mountPlaneACMAY": "{{$FORM.mountPlaneACMAY}}", "mountPlaneACJUN": "{{$FORM.mountPlaneACJUN}}", "mountPlaneACJUL": "{{$FORM.mountPlaneACJUL}}", "mountPlaneACAUG": "{{$FORM.mountPlaneACAUG}}", "mountPlaneACSEP": "{{$FORM.mountPlaneACSEP}}", "mountPlaneACOCT": "{{$FORM.mountPlaneACOCT}}", "mountPlaneACNOV": "{{$FORM.mountPlaneACNOV}}", "mountPlaneACDEC": "{{$FORM.mountPlaneACDEC}}", "mountPlaneKWatts": "{{$FORM.nrelSysCapacityKW}}", "mountPlanePanels": "{{$FORM.mountPlanePanels}}", "mountPlanePanelWatts": "", "mountPlaneTilt": "{{$FORM.mountPlaneTilt}}", "mountPlaneAzimuth": "{{$FORM.mountPlaneAzimuth}}", "mountPlaneACkWhYR": "{{$FORM.mountPlaneACkWhYR}}", "txRebate": "{{$FORM.txRebate}}", "pdRebate": ""}, "complete": "dmxDataBindingsAction('refresh','solarEntryDetails',{});", "success": "dmxDataBindingsAction('refresh','mountingPlanes',{});MM\_callJS('$(\\'body\\').pgNotification({\\n style: \\'bar\\',\\n position: \\'top\\',\\n message: \\'Mounting Plane Added.\\',\\n timeout: 5000,\\n type: \\'success\\'\\n}).show();');"}

);

/\* END dmxDatabaseAction name "mountingPlaneADD" \*/

Roofing:

1. On roofing needs to be sq feet not sq feets.