Dynamic Solar Calculator Planning (Steps 1–5)

Step 1: Inputs

These are things **the user provides**, either by typing or selecting:

Input	Type	Notes
Consumption entries	number	Added via “Add” button; sum gives `totalConsumption`
Selected solar panel	select	Determines `wattsPerPanel`
Annual sun hours	number (optional)	Default 1850h; could be editable
Degradation rate	number (optional)	Default 0.5%; could be editable

---

Step 2: Dependent Calculations

These are **calculated variables** — each depends on other variables:

Variable	Formula / Dependency	Notes
`totalConsumption`	sum of all entries	Input source
`numPanels`	ceil(`totalConsumption / (wattsPerPanel * annualSunHours / 1000)`)	Depends on totalConsumption & panel selection
`systemSize`	`numPanels * wattsPerPanel`	Depends on numPanels & panel
`annualProduction`	`(wattsPerPanel * numPanels * annualSunHours) / 1000`	Depends on numPanels & panel
`coverage`	`(annualProduction / totalConsumption) * 100`	Depends on annualProduction & totalConsumption
`degradedProduction`	`annualProduction * (1 - degradationRate/100)`	Optional, depends on annualProduction & degradationRate

> Optional: later we can add other variables like “Return on Investment”, “Payback time”, etc., but for now, these cover the solar sizing.

Step 3: Event Triggers

We need to **recalculate dependent variables whenever an input changes**:

1. **Consumption entry added** → updates `totalConsumption` → triggers update of all dependent solar variables.
2. **Solar panel selected** → updates `wattsPerPanel` → triggers update of all dependent solar variables.
3. **Optional: Annual sun hours or degradation changed** → triggers update.

   The idea is that any "source" change cascades to all dependent calculations automatically.

Step 4: UI Sketch

We'll need:

1. **Consumption ticket** — shows each entry and total.
2. **Panel selection dropdown** — lets user pick a panel.
3. **Solar data table** — displays all calculated variables dynamically:

Label	Input (readonly or editable)	Unit
# de paneles	input#numPanels	pzas
Watts por panel	input#wattsPanel	W
Tamaño del sistema	input#systemSize	W
Producción anual	input#annualProduction	kWh
Cubre el	input#coverage	%
Producción degradada (optional)	input#degradedProduction	kWh

   Each input gets an **ID** so JS can update it dynamically.

Step 5: Flow Summary

```
User adds consumption → totalConsumption recalculated → updateSolarData()
User selects panel → wattsPerPanel updated → updateSolarData()
Optional: user changes sun hours/degradation → updateSolarData()

updateSolarData():
   1. calculate numPanels
   2. calculate systemSize
   3. calculate annualProduction
   4. calculate coverage
```

```
5. (optional) calculate degradedProduction
6. update table inputs with new values
```

Once this map is clear, coding is mostly implementing `updateSolarData()` and wiring up event listeners, everything else falls into place.