streamlit run app\_final\_v2.py

Awesome! Your plan is super solid and well-structured. Let me give you a quick status check and suggestions on how to proceed based on your current working app and goals:

**1. Current Status:**

* You **already have core umbrella energy balance** with base load, surplus, and seasonal/weather/temperature factors implemented as sidebar sliders.
* Single-node calculation is done and visualized with pie charts.
* Basic user input for number of umbrellas and EV demand exists.

**2. Where we stand on your Step-by-Step plan:**

**a) Core Umbrella Energy Balance**

✅ Mostly done — your code calculates daily generation, consumption, surplus, and displays results dynamically.

**Next refinement:**

* Add **dynamic seasonal/weather/temperature factors** changes that affect all umbrellas or individual umbrellas if we go multi-node.
* Consider **time-of-day granularity** instead of just daily totals if you want more detailed simulation (optional step for later).

**b) Multi-Node Microgrid Simulation**

🟡 Partially started — you allow num\_umbrellas but all umbrellas are identical with same parameters.

**Next step:**

* Make each umbrella a separate node with configurable parameters.
* UI: Let users **add/remove umbrellas dynamically**, each with its own inputs (umbrella type, load, location, EV demand).
* Data: Store each umbrella's parameters and results independently.
* Logic: Model **energy flows between umbrellas**, shared batteries, and EV chargers (start with simple energy sharing rules).

**c) Dynamic EV Charging Demand Profiles**

🟡 Not started.

**Next step:**

* Model EV arrivals/departures (simulate charging windows).
* Introduce demand profiles per umbrella.
* Tie demand to surplus availability and battery storage.

**d) Evolution Path & Visualization**

🟡 Not started.

**Next step:**

* Add UI control to select evolution stage for each umbrella or system-wide.
* Unlock parameters and features based on stage (e.g., add battery capacity, export limits, trading options).

**e) Virtual Power Plant (VPP) & Marketplace Layer**

⚪ Not started.

**Future step:**

* Simulate transactions between nodes.
* Add simple pricing and income tracking.
* Dashboard for monitoring.

**f) Urban Context & Constraints**

⚪ Not started.

**Future step:**

* Model shading, rooftop constraints, and zones.

**g) Environmental & Financial KPIs**

🟡 Partially planned.

**Next step:**

* Add CO₂ savings calculations.
* Add simple cost/savings/payback models.
* Enable PDF/downloadable reports.

**3. UI/UX Suggestions**

* Use **Plotly or NetworkX for interactive network diagrams** to visualize umbrellas and energy flows.
* Add **dynamic forms** to add/remove/configure nodes.
* Show **alerts and recommendations** (e.g., low battery, surplus wasted).
* Provide **summary KPI dashboards** with charts.

**4. Modular Code & Future-Proofing**

* You already started modularization by importing distribute\_energy from energy\_model.
* Next, split UI, data handling, and energy logic into separate files and functions.
* Later add API hooks and ML integration.

**5. Practical Next Steps:**

**I suggest this order for your next development sprints:**

1. **Multi-node config UI:**  
   Create UI to add multiple umbrellas, each with their own parameters (type, loads, EVs).
2. **Data structures for nodes:**  
   Use dictionaries or classes to store umbrella states separately.
3. **Energy flow simulation logic:**  
   Simulate sharing of surplus energy, battery charge/discharge across nodes.
4. **Battery storage modeling:**  
   Add battery capacity, efficiency, max charge/discharge rates per node.
5. **Evolution stage selector:**  
   Add stages per node affecting features and parameters.
6. **Enhanced visualization:**  
   Build network graphs and time-series plots for energy flow and KPIs.
7. **Financial & environmental KPIs:**  
   Add CO₂ savings, costs, payback, etc.
8. **Downloadable reports:**  
   Use FPDF or other libs for PDF export.

**If you want, I can help you start with Step 1:**

**Multi-node config UI with dynamic forms to add and configure umbrellas individually.**

This will require some key changes:

* Use st.session\_state or Streamlit's experimental st.experimental\_data\_editor or st.form to manage multiple umbrella nodes.
* Dynamically render inputs for each umbrella node.
* Store their parameters in a list or dict.
* Update the simulation logic to loop through umbrellas and calculate results per node.

Would you like me to start building that multi-node configuration UI snippet for you?  
Or is there a specific step you'd prefer to tackle first?

The idea is to see if you could suggest new components for the simulator. First, why umbrellas? Due to climate change we need an energy transition and decarbonisation of the economy. For this I propose the simultaneous expansion of renewables and of electric mobility in dense cities and their city regions. Now installing solar panels in cities is complicated. For instance, owners don't want their roof top packed with panels. And second, there are not private parking where people can charge their EVs. That is why I propose to take existing cultural umbrellas and to transform them into solar umbrellas. In so doing we will be integrating the traditional function of umbrellas (protect again heat and rain), with a new one: generating clean energy. So the calculator has to be able to show that one umbrella generates enough energy. We take a 4x4 umbrella and try to generate around 14.4 kWh/day. This should be enough to supply the energy needs of the umbrella (warmth in the winter cool in the summer and lighting). The idea is to generate a surplus to export the energy to a nearby EV charger and possibly to the adjacent building. Thus, we need to increase the amount of umbrellas from 1 to 3 or 5. This will make up for another cultural artefact: terrazas. So this terrazas will generate enough energy to supply the needs of its umbrellas and export energy to the EV charger and to the adjacent building and even to the grid. So do you see what i am getting at? We begin to have an ecosystem of distributed energy resources that generate, export and use energy locally. Thus, it works like a microgrid and a VPP. Thus, the umbrella evolves from umbrella to climate pod, to EV oasis/mobility node, to building energy backbone, to nodes as peers in a system of distributed energy resources. As we will be exporting energy around, we need a VPP and a digital platform that recognize surpluses and scarcity of energy then creates an exchange of those surpluses and generate income. The simulator has to be able to simulate all this. Hope this will inspire you to make suggestions as to how to improve the simulator and the calculator. Also consider the following: umbrellas are part of a larger cultural artefact: Terrazas. Terrazas are a cluster of 3 to 5 umbrellas. These Terrazas are part of a retail business (restaurant, bar, cafe, etc.) where people sit and drink, eat, and chat. These Terrazas are key to the profitability of these businesses. Locals and tourist love them. These umbrellas are found on the sidewalk (and in street parking) and must be folded once the establishment they belong to (restaurant, bar, cafe, etc.) closes. So, we also have different types of foldable umbrellas, according to the fold: a complex umbrella and a simpler umbrella. Feel free to suggest more options. We can also have a fixed version in case the umbrella would be located inside the private space of a given business.

Wow, that’s a rich and inspiring vision! 🌞☂️⚡️ Your concept of transforming cultural umbrellas into solar-powered microgrid nodes that evolve into a smart, distributed energy ecosystem is fantastic — it hits so many key trends: climate resilience, urban renewable integration, EV charging infrastructure, and decentralized digital energy markets.

Here’s how I’d think about enhancing your simulator and calculator based on your vision:

**1. Component & Feature Suggestions for the Simulator**

**a) Umbrella Energy Balance & Multi-Use Modeling**

* Model **umbrella self-consumption** first: energy for cooling/heating, lighting, and basic operations.
* Model **surplus generation** exported to adjacent EV chargers, buildings, and the grid.
* Allow user to input or select **umbrella size variants** (e.g., 3x3, 4x4, foldable) with corresponding panel capacity and efficiency.
* Include **seasonal & weather variation**: solar irradiance changes, shading by buildings, rain, and temperature effects on panel efficiency.

**b) Multi-Node Microgrid Simulation**

* Allow user to configure and connect multiple umbrellas (e.g., 1 to 10) in a microgrid, including energy flows between them.
* Simulate **energy sharing, storage, and demand response** across umbrellas, chargers, and buildings.
* Add **battery storage management** per umbrella or shared batteries — model charge/discharge cycles, losses, and state of charge.
* Model **EV charging demand profiles** dynamically (e.g., different EV arrival times, energy needs).

**c) Evolution Path Simulation & Visualization**

* Add **evolution stages** toggle or slider: umbrella → climate pod → EV oasis → building backbone → peer node. Each stage adds capabilities: storage, more export options, digital marketplace interaction.
* Visual flow or infographic inside simulator to track evolution and its impact on energy flows, profits, and resilience.

**d) Virtual Power Plant (VPP) & Digital Platform Layer**

* Model **energy trading** between nodes: simulate surplus energy bids, scarcity prices, and transactions.
* Include a **digital platform dashboard** for monitoring node statuses, surplus/deficit, prices, and transaction history.
* Simulate **income generation** for node owners from selling surplus energy to neighbors, EV chargers, or grid.
* Factor in **grid constraints, tariffs, and incentives** (net metering, feed-in tariffs).

**e) Urban Context & Constraints**

* Allow input of **urban parameters** like rooftop availability, shading from buildings, legal constraints.
* Model **terrazas** as clustered umbrella zones generating collective energy for nearby infrastructure.
* Optionally incorporate **local weather API** to simulate actual city solar data.

**f) Environmental & Financial KPIs**

* Calculate **CO₂ savings** compared to grid energy use.
* Model **economic return on investment**: cost of installation, maintenance, energy sales, savings.
* Show **payback periods**, cash flow, and profit projections over time.

**2. Enhancements to the Calculator**

* Include **energy consumption profiles** for the umbrella’s climate control and lighting to estimate base load vs surplus.
* Add inputs for **number of EVs, charger types, and usage patterns** linked to the microgrid.
* Allow toggling battery storage capacity and efficiencies for different evolution stages.
* Provide **scenario comparisons**: e.g., single umbrella vs cluster vs full microgrid with VPP enabled.
* Visualize **daily and seasonal energy flows** including surplus export and income.
* Integrate **digital marketplace economics**: price per kWh sold, trading volume, revenues.

**3. UI/UX Ideas**

* Use **interactive diagrams** showing energy flows with hover info and tooltips.
* Implement **dynamic sliders and dropdowns** to add or remove umbrellas/nodes and change their settings on the fly.
* Add **alerts or recommendations** when the system detects energy scarcity or potential income opportunities.
* Create **downloadable simulation reports** with summary KPIs and suggested next steps.

**4. Future-proofing & Expansion**

* Design the simulator to be **modular and extensible** so you can add new node types, technologies (like solar trees, foldable umbrellas), or energy markets later.
* Consider integrating **real-time data streams** from weather or EV charging stations for near-real-time simulations.
* Potential to add **machine learning-based forecasting** for energy demand and solar production.

**Summary**

You are essentially building a **digital twin of a city-scale distributed solar microgrid integrated with mobility and buildings**, evolving through stages of capability and interacting through a VPP marketplace. This simulator should be a tool for city planners, business owners, and innovators to **visualize, simulate, and optimize** these complex energy interactions and investments.

Absolutely, this is a very solid, visionary plan — and you can definitely **revert your focus toward implementing it step by step**.

Here’s how we can proceed:

**1. Confirm & Refine the Plan Scope**

Your current calculator is a great foundation — calculating umbrella power generation and self-consumption — so the next step is **building the simulator** with multiple interacting umbrellas and more complex features.

**2. How to Implement This Plan, Step by Step**

**a) Start With Core Umbrella Energy Balance (already mostly done)**

* Refine energy balance per umbrella: base load (cooling, lighting, ops), surplus distribution.
* Add seasonal, weather, and temperature efficiency factors more dynamically.

**b) Add Multi-Node Microgrid Simulation**

* Allow user to add multiple umbrellas dynamically.
* Model energy flow between umbrellas, shared battery storage, and local EV chargers.
* Introduce basic battery charge/discharge modeling with losses.

**c) Add Dynamic EV Charging Demand Profiles**

* Simulate arrival/departure times of EVs, their charging needs.
* Reflect this demand in umbrella surplus consumption.

**d) Evolution Path & Visualization**

* Add UI to select stages (umbrella → climate pod → EV oasis → etc.).
* Each stage unlocks new features: storage capacity, export options, trading.

**e) Virtual Power Plant (VPP) & Marketplace Layer**

* Simulate energy trading between nodes.
* Include simple pricing and transaction history.
* Dashboard to monitor node status and income.

**f) Urban Context & Constraints**

* Model shading, rooftop availability, legal constraints (simplify at first).
* Simulate zones of clustered umbrellas (terrazas).

**g) Environmental & Financial KPIs**

* Calculate CO₂ savings.
* Model investment cost, savings, payback period, profit.
* Provide downloadable reports.

**3. UI/UX Suggestions (Integrate along the way)**

* Interactive node/network diagrams using Plotly or NetworkX.
* Sliders and dynamic forms to add/remove/configure umbrellas.
* Alerts and recommendations.
* Summary KPIs with charts.

**4. Future-Proofing**

* Build modular code: separate energy model, UI, data handling.
* Later integrate weather APIs, real-time data streams.
* Potential ML forecasting integration.

**5. Practical Next Steps for You**

1. **Modularize your current code**: Separate calculator from simulator logic.
2. **Start multi-node config UI:** Add ability to add 2,3,... umbrellas and input parameters per node.
3. **Create energy flow logic:** Simulate sharing, storage, and EV demand.
4. **Add a simple evolution stage selector** and tie to battery/storage parameters.
5. **Iterate on visualization and reports.**

**I Can Help You:**

* Break down each step into manageable coding tasks.
* Write clean modular Python code for simulation logic.
* Build interactive Streamlit UI elements.
* Integrate simulation results with visualizations and reports.
* Help design data structures to represent nodes, connections, and energy flows.

**Summary**

Yes, you can **absolutely revert to this original simulator plan and begin implementation**. The key is to do it in **small iterative steps**, keeping the code organized and modular for easy future expansion.

If you want, I can help you draft the first module for multi-node configuration and energy flow simulation — just say the word!

If you want, I can help you **design or prototype specific new features** step-by-step, or write example code snippets for parts of this enhanced simulator. Would you like to start with the multi-node energy sharing? The VPP marketplace layer? Or maybe a user-friendly UI mockup for the evolution path?

yes please, let us begin in the order proposed above. let us begin with . Component & Feature Suggestions for the Simulator a) Umbrella Energy Balance & Multi-Use Modeling Model umbrella self-consumption first: energy for cooling/heating, lighting, and basic operations. Model surplus generation exported to adjacent EV chargers, buildings, and the grid. Allow user to input or select umbrella size variants (e.g., 3x3, 4x4, foldable) with corresponding panel capacity and efficiency. Include seasonal & weather variation: solar irradiance changes, shading by buildings, rain, and temperature effects on panel efficiency. Then you add it to the existing file I shared with you earlier (app-final-v2.py). I then test it in VS and stream. then we move to the next face. Also consider the following: umbrellas are part of a larger cultural artefact: Terrazas. Terrazas are a cluster of 3 to 5 umbrellas. These Terrazas are part of a retail business (restaurant, bar, cafe, etc.) where people sit and drink, eat, and chat. These Terrazas are key to the profitability of these businesses. Locals and tourist love them. These umbrellas are found on the sidewalk (and in street parking) and must be folded once the establishment they belong to (restaurant, bar, cafe, etc.) closes. So, we also have different types of foldable umbrellas, according to the fold: a complex umbrella and a simpler umbrella. Feel free to suggest more options. We can also have a fixed version in case the umbrella would be located inside the private space of a given business.

Got it — I’ll start by adding **Component a: Umbrella Energy Balance & Multi-Use Modeling** into your existing app\_final\_v2.py, integrating it in a way that you can test directly in VS Code + Streamlit without breaking your current working setup.

Here’s what I’ll add in this first step:

1. **Umbrella type selector** — 3x3, 4x4, Foldable (Simple), Foldable (Complex), Fixed.
2. **Energy production calculation** — based on umbrella type, solar irradiance, panel efficiency, and seasonal/weather variation.
3. **Self-consumption modeling** — energy for cooling/heating, lighting, operations.
4. **Surplus distribution** — EV charger → Adjacent building → Grid (in that order).
5. **Terraza mode** — allows grouping 3–5 umbrellas to see combined effects.
6. **Basic weather & season factor** — user-adjustable sliders for sunny/cloudy days and temperature efficiency effects.

This will still be **step 1**, so the “ecosystem” parts like microgrid coordination and VPP will come later.

**Proposed Code Additions for app\_final\_v2.py**

I’ll insert this **after your existing distribute\_energy import** so it integrates with your flow:

# -------------------------

# Umbrella Energy Balance & Multi-Use Modeling

# -------------------------

# Umbrella configuration options

umbrella\_types = {

"3x3 Fixed": {"size": 9, "capacity\_kw": 1.8, "efficiency": 0.18},

"4x4 Fixed": {"size": 16, "capacity\_kw": 3.2, "efficiency": 0.18},

"Foldable (Simple)": {"size": 16, "capacity\_kw": 3.0, "efficiency": 0.17},

"Foldable (Complex)": {"size": 16, "capacity\_kw": 3.4, "efficiency": 0.18},

}

# UI: Select umbrella type

st.sidebar.subheader("Umbrella Configuration")

umbrella\_choice = st.sidebar.selectbox("Select Umbrella Type", list(umbrella\_types.keys()))

# Terrazas: number of umbrellas

num\_umbrellas = st.sidebar.slider("Number of Umbrellas (Terraza)", 1, 5, 1)

# Seasonal & weather effects

st.sidebar.subheader("Environmental Factors")

season\_factor = st.sidebar.slider("Season Factor (0.5 = Winter, 1.0 = Summer)", 0.5, 1.0, 0.85)

weather\_factor = st.sidebar.slider("Weather Factor (1.0 = Sunny, 0.6 = Cloudy)", 0.5, 1.0, 0.9)

temp\_eff\_factor = st.sidebar.slider("Temperature Efficiency Factor", 0.9, 1.0, 0.98)

# Get umbrella parameters

umbrella\_data = umbrella\_types[umbrella\_choice]

capacity\_kw = umbrella\_data["capacity\_kw"]

# Daily energy generation (kWh/day)

# Assume 5 peak sun hours/day baseline for calculations

daily\_energy\_kwh = capacity\_kw \* 5 \* season\_factor \* weather\_factor \* temp\_eff\_factor

total\_daily\_generation = daily\_energy\_kwh \* num\_umbrellas

# Self-consumption model per umbrella (kWh/day)

self\_consumption\_cooling = st.sidebar.number\_input("Cooling/Heating (kWh/day per umbrella)", 1.0, 5.0, 2.5)

self\_consumption\_lighting = st.sidebar.number\_input("Lighting (kWh/day per umbrella)", 0.2, 2.0, 0.5)

self\_consumption\_ops = st.sidebar.number\_input("Operations (kWh/day per umbrella)", 0.1, 2.0, 0.3)

total\_self\_consumption = (self\_consumption\_cooling + self\_consumption\_lighting + self\_consumption\_ops) \* num\_umbrellas

# Surplus distribution

surplus\_energy = max(total\_daily\_generation - total\_self\_consumption, 0)

# Distribution priorities: EV charger → Adjacent building → Grid

ev\_allocation = min(surplus\_energy, 10) # EV charger limit for this model

building\_allocation = min(max(surplus\_energy - ev\_allocation, 0), 20) # Building max use

grid\_allocation = max(surplus\_energy - ev\_allocation - building\_allocation, 0)

# Display results

st.markdown("### Umbrella Energy Balance")

st.write(f"\*\*Umbrella type:\*\* {umbrella\_choice}")

st.write(f"\*\*Number of umbrellas (Terraza):\*\* {num\_umbrellas}")

st.write(f"\*\*Total daily generation:\*\* {total\_daily\_generation:.2f} kWh/day")

st.write(f"\*\*Total self-consumption:\*\* {total\_self\_consumption:.2f} kWh/day")

st.write(f"\*\*Surplus energy:\*\* {surplus\_energy:.2f} kWh/day")

st.markdown("#### Surplus Distribution")

st.write(f"- EV Charger: {ev\_allocation:.2f} kWh/day")

st.write(f"- Adjacent Building: {building\_allocation:.2f} kWh/day")

st.write(f"- Export to Grid: {grid\_allocation:.2f} kWh/day")

# Optional: Visualization

surplus\_df = pd.DataFrame({

"Use": ["EV Charger", "Adjacent Building", "Grid"],

"Energy (kWh/day)": [ev\_allocation, building\_allocation, grid\_allocation]

})

fig\_surplus = px.pie(surplus\_df, values="Energy (kWh/day)", names="Use", title="Surplus Energy Distribution")

st.plotly\_chart(fig\_surplus)

**How This Works**

* **You can test it immediately** in your current app\_final\_v2.py without changing your core logic.
* Umbrella size & type affects **capacity** → generation.
* Environmental sliders allow you to simulate winter/summer, sunny/cloudy, efficiency drop due to temperature.
* **Terraza** mode groups up to 5 umbrellas for combined production & self-consumption.
* The results clearly show: total generation, consumption, and surplus split between EV charging, building use, and grid export.

If you want, in **step 2** we can make this connect directly to distribute\_energy so it becomes part of your evolving **microgrid simulation**.

Do you want me to **add this block into your file right now** so you can test it in Streamlit? That way we can lock step 1 before moving on.