Here’s a copy-paste spec Claude can implement directly for the **Projects → Edit** flow. It covers UI layout, validation, upload parsing, JSON schemas, mapping to the permutation engine, and save logic — including the optional monthly schedule.

**Projects Tool — Edit Modal (Sponsor Input)**

**Trigger**

* Open when user clicks **Edit** on a project tile.
* Route: GET /api/projects/:id → prefill form.
* Save: POST /api/projects/:id (upsert).

**UI Layout (sections & fields)**

**1) Project Metadata**

* projectTitle (text, required)
* locationCountry (text, required)
* status (select: Draft | Final | Submitted; default Draft)
* currency (select: EUR | USD | GBP | JPY | AED; required)
* grossITLoadMW (number, required, min 0.1, step 0.01)
* pue (number, required, min 1.0, max 2.5, step 0.01)
* grossMonthlyRent (number, optional, ≥ 0; units: currency/month)
* useOpexPercent (boolean, default true)
* opexPercent (number, optional, 0–1; example: 0.20 for 20%)
* constructionStart (string, optional, format YYYY-MM; example 2025-05)
* constructionDurationMonths (integer, optional, 0–40)

**2) Upload OR Manual Entry (tabs)**

**Tab A — Upload (preferred)**

* File input .xlsx (accept):
  + Project\_Sponsor\_Input\_Template.xlsx OR
  + Project\_Sponsor\_Input\_Template\_PLUS\_Schedule.xlsx
* On upload:
  + Parse sheets:
    - Engine\_Inputs (always)
    - Schedule\_Optional (if present/non-zero)
    - Monthly\_Rollup (derived; read if schedule exists)
  + Display a read-only preview: CapEx, Land, Ops, Fees, Contingency, Grand Total, and (if present) the monthly curve totals.

**Tab B — Manual Entry**

* **Build (CapEx) — amounts (EUR):**
  + build.architectural
  + build.civilStructural
  + build.substations
  + build.undergroundUtilities
  + build.crusaderModules
  + build.bess
  + build.mainContractorInstallation
  + build.photovoltaicSystem
  + build.landscaping
  + build.preliminariesGeneral
  + build.mainContractorMargin
* **Land & Infra — amounts (EUR):**
  + land.landPurchase
  + land.additionalLand
  + land.gridConnection (number OR set land.gridIncluded = true)
  + land.stampDutyPercent (default 0.10) and land.stampDutyFixed (fixed overrides %)
* **Operations — amounts (EUR):**
  + ops.operationsTeam
  + ops.ose
  + ops.marketing
* **Fees — amounts (EUR):**
  + fees.dm
  + fees.management3pct
* **Contingency — percent (decimal):** contingencyPercent (default 0.10; applied to Build + Land + Grid + Ops; excludes stamp and contingency itself)

The UI shows running **roll-ups** (read-only):

* buildTotal, opsTotal, feesTotal, landTotal (land + addl + grid + stamp), contingencyAmount, grandTotal.

**3) Optional Monthly Schedule**

* If the uploaded file contains a schedule (non-zero in Schedule\_Optional), show a toggle **“Use monthly schedule for cash curve”** (default ON).
* If manual path is used, this section is hidden (schedule is optional upload-only in v1).

**4) Actions**

* **Cancel**
* **Save Changes** (validates + persists + emits recalculation event for permutation engine)

**Validation Rules**

* Currency required.
* grossITLoadMW > 0.
* pue in [1.00, 2.50].
* If useOpexPercent = true then opexPercent ∈ [0, 1]; else ignore opexPercent.
* Non-negative numbers everywhere; empty = treated as 0.
* land.stampDutyFixed overrides land.stampDutyPercent if > 0.
* gridIncluded = true forces gridConnectionEffective = 0.

**Derived Calculations (identical to the templates)**

buildTotal = sum(all build.\*)

gridConnectionEffective = land.gridIncluded ? 0 : (land.gridConnection || 0)

stampDutyAmount = land.stampDutyFixed > 0

? land.stampDutyFixed

: (land.stampDutyPercent || 0) \* ((land.landPurchase||0) + (land.additionalLand||0))

opsTotal = sum(ops.\*)

feesTotal = sum(fees.\*)

contingencyBase = buildTotal + (land.landPurchase||0) + (land.additionalLand||0) + gridConnectionEffective + opsTotal

contingencyAmount = (contingencyPercent || 0) \* contingencyBase

landTotal = (land.landPurchase||0) + (land.additionalLand||0) + gridConnectionEffective + stampDutyAmount

grandTotal = buildTotal + landTotal + opsTotal + feesTotal + contingencyAmount

grossAnnualIncome = (grossMonthlyRent || 0) \* 12

opexAnnual = useOpexPercent ? (opexPercent || 0) \* grossAnnualIncome : 0

**JSON Schema — Save Payload**

**A) Minimal (manual or roll-up only)**

{

"projectId": "<uuid>",

"meta": {

"projectTitle": "Portugal Hyperscale",

"locationCountry": "Portugal",

"status": "Draft",

"currency": "EUR",

"grossITLoadMW": 83.33,

"pue": 1.25,

"grossMonthlyRent": 0,

"useOpexPercent": true,

"opexPercent": 0.20,

"constructionStart": "2025-05",

"constructionDurationMonths": 24

},

"rollups": {

"capexCostPriceEUR": 742197829,

"landPurchaseFeesEUR": 120000000 + 1000000 + 0 + 21500000,

"operationsAnnualEUR": 0,

"contingencyEUR": 73129269,

"feesTotalEUR": 343, // example, fill real

"grandTotalEUR": 838360734

},

"schedule": null

}

**B) With Monthly Schedule (preferred if uploaded)**

{

"projectId": "<uuid>",

"meta": { ...same as above... },

"rollups": { ...same as above... },

"schedule": {

"months": ["M1","M2","M3","M4", "...", "M24"], // up to M40

"buckets": ["BUILD","LAND","OPS","FEES","CONTINGENCY","TOTAL"],

"byBucket": {

"BUILD": [1082116,1082116,1082116,10415220, "..."],

"LAND": [0,0,0,12000000,"..."],

"OPS": [12821,12821,12821,12821,"..."],

"FEES": [0,0,0,2727697,"..."],

"CONTINGENCY":[109494,109494,109494,252735,"..."],

"TOTAL": [1204430,2408860,3613290,22680776,"..."]

}

}

}

**Mapping to Permutation Engine (single source of truth)**

The **engine input object** your calc service expects:

{

"currency": "EUR",

"grossITLoadMW": 83.33,

"pue": 1.25,

"capexCostPriceEUR": <buildTotal>,

"landPurchaseFeesEUR": <landTotal>,

"operationsOpexAnnualEUR": <opexAnnual>, // 0 if useOpexPercent=false or no rent

"grossMonthlyRentEUR": <grossMonthlyRent>, // optional

"contingencyEUR": <contingencyAmount>,

"feesTotalEUR": <feesTotal>,

"grandTotalProjectEUR": <grandTotal>,

"constructionStart": "2025-05",

"constructionDurationMonths": 24,

"monthlyCurve": { // optional

"months": [...],

"totalPerMonth": [...] // use schedule.byBucket.TOTAL, else null

}

}

**Upload Parser (pseudocode)**

async function parseSponsorXlsx(file: Buffer): Promise<{meta, rollups, schedule|null}> {

const wb = await readXlsx(file);

// Prefer Engine\_Inputs if present

const eng = readSheetAsDict(wb, "Engine\_Inputs");

const pm = readSheetAsDict(wb, "Project\_Meta"); // for meta fields (currency, load, pue, etc.)

const meta = {

projectTitle: pm.Project\_Title || "",

locationCountry: pm.Location\_Country || "",

status: pm.Status || "Draft",

currency: eng.Currency || pm.Currency,

grossITLoadMW: num(eng.GrossITLoad\_MW || pm.Gross\_IT\_Load\_MW),

pue: num(eng.PUE || pm.PUE),

grossMonthlyRent: num(eng.GrossMonthlyRent\_EUR\_\_optional\_ || pm.Gross\_Monthly\_Rent),

useOpexPercent: str(pm.Use\_OPEX\_Percent).toUpperCase() === "YES",

opexPercent: num(pm.OPEX\_Percent),

constructionStart: eng.Construction\_Start\_YYYY\_MM || pm.Construction\_Start\_YYYY\_MM || null,

constructionDurationMonths: int(eng.Construction\_Duration\_Months || pm.Construction\_Duration\_Months)

};

const rollups = {

capexCostPriceEUR: num(eng.CapEx\_CostPrice\_EUR),

landPurchaseFeesEUR: num(eng.LandPurchaseFees\_EUR),

operationsAnnualEUR: num(eng.Operations\_OPEX\_Annual\_EUR\_\_if\_used\_),

contingencyEUR: num(eng.Contingency\_EUR),

feesTotalEUR: num(eng.Fees\_Total\_EUR),

grandTotalEUR: num(eng.Grand\_Total\_Project\_EUR)

};

// Optional schedule

let schedule = null;

if (sheetExists(wb, "Schedule\_Optional")) {

const S = readSheet(wb, "Schedule\_Optional"); // rows: line items, cols: M1..M40

if (hasAnyNonZero(S, ["M1","M2",/\*...\*/])) {

const MR = readSheet(wb, "Monthly\_Rollup"); // rows: buckets, cols: M1..M40 + Total

const months = headerMonths(MR); // ["M1"...] up to last non-empty month

schedule = {

months,

buckets: ["BUILD","LAND","OPS","FEES","CONTINGENCY","TOTAL"],

byBucket: {

BUILD: rowToArray(MR, "BUILD", months),

LAND: rowToArray(MR, "LAND", months),

OPS: rowToArray(MR, "OPS", months),

FEES: rowToArray(MR, "FEES", months),

CONTINGENCY: rowToArray(MR, "CONTINGENCY", months),

TOTAL: rowToArray(MR, "TOTAL", months)

}

};

}

}

return { meta, rollups, schedule };

}

**Save Handler (pseudocode)**

// POST /api/projects/:id

async function saveProject(req, res) {

const body = req.body; // or parsed XLSX result

// 1) Validate core fields

validate(body.meta.currency in ["EUR","USD","GBP","JPY","AED"]);

validate(body.meta.grossITLoadMW > 0);

validate(1.0 <= body.meta.pue && body.meta.pue <= 2.5);

if (body.meta.useOpexPercent) validate(0 <= body.meta.opexPercent && body.meta.opexPercent <= 1);

// 2) Persist project + latest sponsor payload

await db.projects.update(req.params.id, { meta: body.meta, rollups: body.rollups, schedule: body.schedule });

// 3) Build engine input object

const engineInput = {

currency: body.meta.currency,

grossITLoadMW: body.meta.grossITLoadMW,

pue: body.meta.pue,

capexCostPriceEUR: body.rollups.capexCostPriceEUR,

landPurchaseFeesEUR: body.rollups.landPurchaseFeesEUR,

operationsOpexAnnualEUR: body.rollups.operationsAnnualEUR || 0,

grossMonthlyRentEUR: body.meta.grossMonthlyRent || 0,

contingencyEUR: body.rollups.contingencyEUR || 0,

feesTotalEUR: body.rollups.feesTotalEUR || 0,

grandTotalProjectEUR: body.rollups.grandTotalEUR || 0,

constructionStart: body.meta.constructionStart || null,

constructionDurationMonths: body.meta.constructionDurationMonths || null,

monthlyCurve: body.schedule ? {

months: body.schedule.months,

totalPerMonth: body.schedule.byBucket.TOTAL

} : null

};

// 4) Emit event → permutation engine

await queue.publish("permutation.requested", { projectId: req.params.id, engineInput });

return res.status(200).json({ ok: true, engineInputPreview: engineInput });

}

**Error States (user-friendly)**

* Invalid or missing currency / IT Load / PUE → inline error on those fields.
* Uploaded file missing Engine\_Inputs → “Template not recognized. Please use the provided template.”
* Schedule present but malformed → “Monthly schedule could not be read. Roll-ups are saved; schedule ignored.”
* Numbers outside bounds → show range hints.

**Notes for Claude**

* Reuse the same field IDs across UI ↔ API ↔ DB to minimize mapping.
* When both **upload** and **manual** are used in one session, **upload wins**; show a banner: “Values populated from upload. You can switch to Manual to override.”
* Persist the original file (blob storage) with a checksum for audit.
* All currency values are stored in **EUR** (as per template); if currency != EUR, keep currency code and let downstream FX module normalize.

This is everything needed to wire the **Edit Project** flow, including optional schedule support and a clean, deterministic mapping into the permutation engine.