

**Project Name:**

***Data Alchemist: Forge Your Own AI Resource‑Allocation Configurator***

**Product Vision**

Modern resource allocation teams drown in messy spreadsheets spread across clients, workers & tasks. The Data Alchemist cleans, validates, corrects and enriches this data with AI:

* Upload raw CSV/XLSX files → get instant AI‑powered parsing & fixes
* Inline edit & get real‑time validations with visual feedback
* Type business rules in plain English → AI turns them into JSON
* Use sliders to set what matters most (e.g. priority vs cost)
* Export clean data + rules config → ready for the downstream allocator

**Built for non‑technical users**: no code, no formulas, only clicks & natural language.

**Tech Stack**

| **Layer** | **Choice** |
| --- | --- |
| Framework | Next.js (TypeScript) |
| UI components | shadcn/ui + Tailwind CSS |
| Data grid | TanStack Table / react-data-grid (editable, validation) |
| XLSX/CSV parsing | SheetJS |
| AI / NLP | OpenAI (via API) or local LLM (o4‑mini) |
| State mgmt | React context + Zustand (for rule configs & weights) |
| Backend (if needed) | Next.js API routes |
| Validation | Custom logic + AI‑driven checks |

**Project Folder Structure**

bash

CopyEdit

/pages

/index.tsx ← Main upload + grid page

/rules.tsx ← Rule builder page

/prioritize.tsx ← Prioritization page

/components

DataGrid.tsx

UploadSection.tsx

ValidationPanel.tsx

RuleBuilder.tsx

NaturalLanguageBar.tsx

/lib

aiParser.ts ← AI header mapping

validators.ts ← Core + AI validators

ruleEngine.ts ← Natural language → rules

/public/samples

clients.csv, workers.csv, tasks.csv

**Features & AI plan**

**1. Data ingestion & AI parsing**

* Upload 3 CSV/XLSX files: clients, workers, tasks
* SheetJS parses them into JSON
* AI header mapper: if columns are misnamed (“client name” vs “ClientName”), AI suggests correct mapping (show confirmation UI)
* Show each entity in an **editable data grid** (fix typos, fix JSON, adjust lists inline)

**2. Validations**

* On upload & every edit: run validations
* Highlight cells/rows with issues (red border, tooltip with message)
* Validation summary panel listing:
  + Missing required columns
  + Duplicate IDs
  + Malformed lists
  + Out-of-range values
  + Broken JSON
  + Unknown references
  + Circular co-run groups
  + Conflicting rules
  + Overloaded workers
  + Phase-slot saturation
  + Skill coverage matrix
  + Max concurrency feasibility

**AI-driven validations:**

* Detect patterns: “Duration=0” likely wrong → suggest fix
* Suggest missing required fields based on data distribution
* Natural language search: “Show me all tasks longer than 2 phases”

**3. Natural Language Data Retrieval & Modification**

* Simple search box → type:

“All clients with PriorityLevel >3 requesting task T12”

* AI parses & filters table view
* Next step: allow AI suggestions like “Fix broken JSON in row 3?” (click to apply)

**4. Rule Input UI**

* Visual rule builder:
  + Select tasks → create co-run rule
  + Choose worker group + max slots → load-limit rule
  + Select task + preferred phases → phase-window rule
  + Enter regex → pattern-match rule
* Rules shown in table; can edit/delete
* Generate rules.json on click

**AI natural language → rule:**

“Tasks T12 and T14 must always run together” → adds co-run rule  
“Limit sales team to 2 slots per phase” → adds load-limit rule

**5. Prioritization & Weights**

* Sliders to set importance: PriorityLevel, task fulfillment, fairness, cost
* Drag & drop to re-order criteria
* Preset profiles (Fastest Finish, Balanced, Cost Saver)
* Export button → gets:
  + Cleaned clients.csv, workers.csv, tasks.csv
  + rules.json with prioritization metadata

**6. X‑Factor AI features**

* AI detects conflicting rules → “Precedence override?” suggestion
* AI suggests new rules:  
  “Tasks T4 & T7 always appear together → add co-run?”
* AI error correction: auto‑fix broken JSON, recommend valid ranges

**UX & UI**

| **Page** | **Key Components** |
| --- | --- |
| Home | Upload section, AI header mapper, data grids, validation panel |
| Rules | Visual rule builder, natural language input, rules table |
| Prioritize | Sliders, drag & drop ranking, presets, export button |

All in a **clean, minimalist UI**: rounded cards, soft shadows, inline edit icons, success/error banners.

**Sample Data Strategy**

**Include in /public/samples:**

* Realistic & messy sample files:
  + Misnamed columns (“client priority”)
  + Missing fields
  + Bad JSON in AttributesJSON
  + Duplicate IDs
* Edge cases: out‑of‑range Duration, unknown TaskIDs

**Milestones & Timeline**

| **Milestone** | **Features** | **Est. time** |
| --- | --- | --- |
| M1 | Upload, AI parser, data grid, core validators, natural language retrieval | 1 day |
| M2 | Rule builder UI, natural language → rules, prioritization sliders & export | 1 day |
| M3 | AI suggestions, auto-corrections, advanced validations, UX polish | 1 day |

Final: Deploy to Vercel & record optional X-factor demo (≤120s)

**Deployment**

* Deploy to Vercel (npm run build && npm run start)
* Add README:
  + App link
  + Features
  + Sample data guide
  + How to run locally

**Summary**

Data Alchemist combines **smart parsing, real‑time validation, AI‑powered natural language control & rule generation**, all wrapped in a friendly Next.js web-app. It’s built so even non‑tech users can **fix messy data, define complex rules & prepare clean configs** with just clicks.