# **Gyld Mini Take-Home — Team Re-Assignment (≈2 hours)**

### **Context**

Gyld runs on seasons. At the start of each season, players can be sorted into new teams. These teams become part of the streamer’s community identity for the season, competing together in events and rituals.

When we reshuffle, we need the outcome to feel balanced and make sense to the community. Teams should be roughly even in size, and the distribution of players should reflect engagement in a way that won’t feel arbitrary if players compared notes.

Your job is to design a simple system to reassign players into **T** new teams for a new season. The assignment should be deterministic and reproducible (i.e. same input → same output). How you define “balanced” is up to you, but the output should look reasonable if shown to real players.

### **Goal**

Reassign players into **T teams** for a new season so teams are balanced and the outcome feels reasonable to the community.

### **What you’ll build (timeboxed to ~2 hours)**

A script/program (TypeScript/Node preferred) that:

* Outputs a mapping of player\_id → new\_team.
* Prints a short per-team summary (team size + a few useful aggregates of your choice).

Include a **brief README** explaining:

* How to run it
* Your approach in plain language
* Key tradeoffs you made due to the time limit
* What you would add if you had more time

### **Constraints**

* Keep team sizes roughly similar.
* Make deterministic choices for ties.
* Aim for an outcome that would look balanced and make sense if shown to the community. (Intentionally broad—interpret as you see fit.)

### **Data**

#### **Level A (required)**

You’ll receive a single dataset of players with fields such as:

* player\_id
* current\_team\_id
* current\_team\_name
* historical\_events\_participated (unique events joined)
* historical\_event\_engagements (total interactions across events)
* historical\_points\_earned
* historical\_points\_spent
* current\_total\_points (always ≤ historical earned)
* historical\_messages\_sent (public chat)
* days\_active\_last\_30
* current\_streak\_value
* last\_active\_ts (unix seconds)

**Task:** Using only this Level A dataset, produce your new team assignment and summary.

#### **Level B (optional, advanced)**

We’ll also provide raw tables (separate CSVs) you may use if you prefer to compute your own aggregates:

* **events** — event participation rows (player\_id, ts, event\_id, points\_used, engagement\_kind, …)
* **spend** — item purchases and consumption (player\_id, ts, item\_id, point\_purchase\_value, is\_consumable, is\_consumed, …)
* **messages** — chat messages (player\_id, ts, text\_length, is\_message\_reply, …)

**Optional:** If you choose, derive your own features from these tables instead of or in addition to Level A fields. Note any assumptions you made in the README.

⚠️ Important: **Level A is sufficient to complete the exercise.** Level B is there only if you want to dig deeper or show how you would approach modeling features from raw logs. Attempt it only if you think it strengthens your approach within the ~2 hour limit.

### **Deliverables**

* Code repo (or zip) with a one-command run (npm start, make run, etc.)
* Output: player\_id → new\_team assignments and a per-team summary
* README with:
  + Run instructions
  + Your approach & tradeoffs
  + Any assumptions (esp. if you used Level B)
  + “If I had more time, I would…”
  + Plain-text or Markdown README is fine

*No UI or polish expected.*

### **What we look for**

* Clear, pragmatic code structure
* Sensible feature modeling and normalization
* Deterministic, balanced assignment logic
* Thoughtful tradeoffs under a time limit
* (Optional) If you used Level B, clean/simple aggregation with explained assumptions

### **Submit**

Share a GitHub link (preferred) or a zip file. We’ll schedule a 30-minute walkthrough.