Production Planner – Product Requirements Document (PRD)

Product: Production Planning & UPH Intelligence Tool\ **Owner:** Atlas Pet Company Ops/Engineering\ **Integrated System:** Fulfil.io (ERP)\ **Version:** 1.0

---

1. Problem & Goals

Operations spends excessive time manually assigning operators and validating time estimates. Historical performance (UPH) is inconsistent across views, and outlier data skews averages. We need a single application that:

1. Computes **accurate, consistent UPH** for every Routing + Operation + Work Center + Operator combination.
2. **Forecasts hours to complete** remaining work on all open MOs/WOs.
3. Provides **fast assignment workflows** (Manual, Auto-Assign AI, Actual) while **enforcing constraints**.
4. Surfaces **anomalies** in cycle data and guides users to fix them in Fulfil.
5. Uses **one canonical data pull** from Fulfil's `search_read` endpoint and modern pagination/filtering.

Success = <3 min to fully assign a week of work, <2% variance between forecast and actual hours, anomaly rate <1% after clean-up.

---

2. Users & Use Cases

- **Production Planner** – builds weekly plans, runs Auto-Assign, resolves conflicts/outliers.
- **Production Lead / Floor Manager** – monitors workload, locks Actual assignments post-run, triages anomalies.
- **Operators** – view workload summaries, ensure assignments respect availability.
- **Ops/Analytics** – audit UPH trends, adjust windows, verify consistency.

---

3. Scope

In-Scope

- Grid UI with three assignment modes and color semantics.
- UPH computation engine + cache.
- AI Auto-Assign (OpenAI) with Try Again / Clear All.
- AI Anomaly Detection toggle, red-pill highlighting, Fulfil deep links.
- Operator Settings (UPH table, toggles, capacity fields).

- Fulfil API proxy/services, pagination (`offset`, `limit`), ordering, filter builder.

Out of Scope (V1)

- AI-optimized routing resequencing.
- Mobile-first UI.
- Updating Fulfil Work Orders with Operator Assignments

---

4. Functional Requirements

4.1 Data Integration

Canonical Work-Cycle Pull\ `PUT /api/v2/model/production.work/search_read`

```
{
  "filters": [["state", "=", ["done", "finished"]]],
  "fields": [
    "id",
    "operator_rec_name",
    "rec_name",
    "production",
    "work_center_category",
    "work_operation_rec_name",
    "production_work_cycles_duration",
    "production_work_cycles_id",
    "work_cycles_work_center_rec_name",
    "state",
    "production_routing_rec_name",
    "production_quantity",
    "create_date",
    "production_planned_date",
    "production_priority"
  ],
  "offset": 0,
  "limit": 500,
  "order": [["create_date","ASC"]]
}
```

- Use `offset` / `limit` for pagination; loop until empty.
- `order` optional but recommended for deterministic processing.

Other Endpoints

- Open WOs (grid): `GET /api/v2/model/production.work?state=request,draft,waiting,assigned,running&fields=...`

• MO meta: `GET /api/v2/model/production.order?id=`

4.2 UPH Algorithm (Canonical)

Process at a glance

```
WO Cycles → WO Duration → WC Duration (per MO) → MO UPH → Category
Roll-up → Historical Avg → Cache
```

Step 1 – Gather Cycles

Pull all completed cycles with `PUT /production.work/search_read` (state = done/finished) for the selected window (7, 30, 180 days).

Step 2 – WO Duration

For each **Routing + Operation + Work Center + Operator + WO** combination, sum all cycle durations (seconds):

```
total_wo_duration_sec = Σ(duration_sec for cycles in that WO/combination)
```

Step 3 – WC Duration per MO

If a single MO has multiple WOs in the same Work Center, add their totals and convert to hours:

```
total_wc_duration_hrs = (Σ total_wo_duration_sec for MO & WC) / 3600
```

Step 4 – MO-level UPH

```
UPH_MO = MO_quantity / total_wc_duration_hrs
```

Store this for each Routing + Operation + Work Center + Operator.

Step 5 – Category Roll-up

Map Work Centers → **Cutting / Assembly / Packaging** (Rope & Sewing & Embroidery → Assembly). Average the MO-level UPHs that belong to each category for that operator/combination.

Step 6 – Historical Average & Cache

Average `UPH_MO` values across the chosen window and cache using the key:

```
(Routing, Operation, Work Center, Operator, Window)
```

Expose the same cached numbers to both the Planner grid and the UPH table.

4.3 Assignment Modes & UI Rules

Mode	Visual	Editable?	Source of Truth
Manual	Operator name **green**	Yes	User selection in grid
Auto-Assign (AI)	Operator name **blue**	Yes (override allowed)	AI result persisted to WO
Actual	Grey/black, dropdown locked	No	Completed Work Cycles (Fulfil)

- If MO/WO already has a completed cycle, lock to Actual.
- Dropdown shows only operators with UPH for that combo and not over capacity.

4.4 Auto-Assign (AI)

Steps:

1. **Prep:** Gather unassigned WOs, operator constraints (hours, toggles), UPH map.
2. **Prompt OpenAI:** Optimize assignments to minimize total hours & idle time under constraints.
3. **Apply:** Write assignments back, color blue, update workload cards.
4. **Controls:** Global/row/column **Try Again**, **Clear All**.
5. **Explainability:** Tooltip with chosen UPH, hours remaining, constraint checks.

4.5 AI Anomaly Detection

- Toggle on UPH/Analytics page.
- Detection: Median + IQR per cohort (fallback z-score >3). Outliers excluded from averages.
- UI: Red-outline pill on UPH value, tooltip “Anomaly – excluded from avg”.
- Modal: List of anomalous MOs, comparator MOs (same product & ±20% qty), links to `#/model/work.cycle/<id>` in Fulfil.
- Banner: “⚠️X anomalies excluded. Review in Fulfil.”

4.6 Operator Settings

- Table: UPH per combo, enable/disable toggles for WC/Operation/Routing, Max Weekly Hours, Schedule %.
- Effective capacity = max_hours × schedule %.
- Persist via PUT on operator model or internal DB.

4.7 Validation & Errors

- Prevent over-capacity assignments.

- Warn when UPH missing → “Data Missing”.
- Lock WOs when state=done.
- Robust unit tests for math, filters, capacity, anomaly pipeline.

---

5. Non-Functional Requirements

- **Performance:** Anomaly detection <1s for 10k MOs; grid loads in <2s with pagination.
- **Reliability:** Nightly job to refresh cache; manual refresh button.
- **Security:** API keys stored server-side; no keys in client.
- **Observability:** Log auto-assign decisions, anomaly counts, cache misses.

---

6. Acceptance Criteria & QA

Area	Test	Pass Condition
UPH Parity	Same numbers in Planner & UPH page for same window	100% match
Anomaly Flag	MO133475 (25,200 UPH) flagged	Flag + excluded avg
Toggle	Switching anomaly toggle updates averages instantly	<250ms UI refresh
Capacity	Cannot save assignment if exceeds capacity	Error + block
Endpoint Contract	search_read filter/fields/limit/offset honored	200 OK + schema match
Performance	Detection job <1s for 10k MOs	Timer logs

---

7. Delivery Checklist

- **Data layer** uses only `PUT /api/v2/model/production.work/search_read` for work-cycle pulls (supports `offset`, `limit`, `order`). No legacy `GET /work.cycles` calls remain.
- **Endpoint coverage**: 2.1 (open WOs), 2.3 (MO meta), 2.5 (assign operator) implemented with the exact `fields` lists and validated params.
- **UPH math** matches §4.2: WO sum → WC per-MO hours (sec→hr) → MO-level UPH → category roll-up → cache by Routing+Operation+WC+Operator+Window.
- **Category mapping** applied everywhere (Cutting / Assembly / Packaging; Rope & Sewing & Embroidery → Assembly).
- **Planner & Analytics parity**: both read from the same service/cache and return identical UPH for identical windows & toggles.
- **Auto-Assign AI** shipped with Try Again / Clear All, constraint enforcement, and blue font for AI rows.
- **Manual / AI / Actual visuals**: green / blue / grey (locked) implemented in grid; Actual locked when cycles exist.

- **Operator Settings page**: UPH table, enable/disable toggles, Max Hours, Schedule %, persisted correctly.
- **AI Anomaly Detection**: toggle, median/IQR (z-score fallback), red-pill UI, banner count, modal with comparator MOs + Fulfil deep links.
- **Validation & error states**: capacity breaches blocked, missing UPH flagged “Data Missing”, invalid combos warned.
- **Caching & jobs**: nightly/cron (or on-demand) recompute documented; TTL and invalidation strategy defined.
- **Tests in CI**: parity, anomaly, toggle, capacity, performance (<1s for 10k MOs), endpoint contract tests.
- **Docs & env vars**: README updated; `.env` (`FULFIL_API_URL`, `FULFIL_API_KEY`) + deployment/runbook.
- **Telemetry/Logging**: anomaly counts, auto-assign outputs, cache misses captured; dashboards or logs accessible.
- **Accessibility & responsiveness**: grid usable at common breakpoints; keyboard navigation and contrast checked.

---

8. Future Roadmap (Post-V1)

- Optimized routing/sequencing.
- Predictive staffing recommendations.
- Cross-product batching optimization.
- Operator skill inference from partial data using ML.

Use this PRD as the single source of truth for development and QA.