PRD — Modisoft Demand & Sales Forecast

Architecture: Hybrid (LightGBM + GPT-4 mini)

Owner: Ebuka (AI PM) **Design Partners:** Ketan (UI/UX), Matellio Ganesh (Developer)
Last updated: 2025-11-21

1. Product Summary

A simple, owner-friendly module that forecasts **revenue (before discounts & promotions)** and **units**, then turns those forecasts into 1-click actions to prevent stock-outs and reduce waste.

- **LightGBM** generates baseline forecasts per store×SKU×day.
- **GPT-4 mini** parses messy inputs (promo text, local events), composes scenario **lifts** with guardrails, writes chart callouts & AI Insights, and routes data to fallbacks when inventory is unreliable.
- The UI adapts to **business type** (Convenience, Restaurant, Grocery/Retail, Liquor). Restaurants are handled in a separate module.
- Users can simulate promotions/price changes and see the impact instantly.
- **Ask Sunny** enables on-demand Q&A.

Why: Reduce stock-outs (lost sales), lower waste, and free working capital while keeping the experience legible for a non-technical 50–65-year-old owner.

2. Goals & Non-Goals

Goals

1. Accurate, plain-English forecasts for revenue (pre-discount) & units.
2. Actionable widgets by business type: **Refill now (6h)**, **48h Category Risk**, **Weekend Run-up**, **Bundles**, **Top Re-order**.

3. Scenario editing (promo/price) with **sub-second** visual updates.
4. Fully **responsive** across desktop/laptop/iPad/tablet/mobile and **Chrome/Safari**.
5. Low-ops: nightly training + hourly refresh, auto data-health checks.
6. Cost discipline: batch with **GPT-4 mini**; large model only for Sunny chat.

Non-Goals

- Exposing EOQ/service-level math or ML diagnostics to users.
- Accounting/PO workflows beyond simple “Add to PO”.

3. Personas

- **Owner/Manager (primary):** wants “what’s coming and what to do today”.
- **Ops Lead / Multi-unit (secondary):** compares stores, ensures execution.
- **Internal Analyst (backstage):** monitors accuracy, cost, and data health.

4. Success Metrics (Business)

- Stock-outs ↓ **12%** vs control within 4 weeks.
- Waste ↓ **8%** (perishables where available) within 8 weeks.
- Promo attach/basket ↑ **5%** in affected categories.
- WAU: **>70%** owners view 3+ days/week.
- P95 dashboard load **<2s**; widget API **<600ms**; daily AI batch < **$0.02/store**.

5. User KPIs (with tooltips)

1. **Money coming in** — gross sales (before discounts & promos) for selected window.
2. **Units we’ll sell** — total items expected.
3. **Today vs typical** — vs normal same-weekday (%).
4. **Promo / price boost** — net lift this period (%).
5. **Weather & events impact** — net lift this period (%).

6. **Data Health (fixes)** — anomalies to review (negative on-hand, stale counts, missing SKU meta, etc.).

"Before discounts & promotions" must appear beneath the page title. Tooltips explain each KPI plainly.

6. Experience & Interaction

6.1 Controls

- Forecast window: **7 / 14 / 28 days**.
- Store selector: **All Stores** or multiple stores (by vertical).
- Filters: **Department** dropdown, **Item** search.
- **Explain changes** toggle overlays chart callouts.
- **Ask Sunny** button (opens chat with context payload).

6.2 Charts

- **Revenue forecast (before discounts)** — line, Actual vs Forecast, annotations when Explain is on.
- **Units forecast** — bars, Actual vs Forecast.
- Axis labels are large; hover tooltips show simple numbers.

6.3 Business-Type Widgets

- **Convenience:**
    - **Fast-Mover Refill (Now → +6h)** — columns: *Expected (next 6h)*, *On-shelf*, *Backroom*, *Refill now*, *Print refill list*.
    - **Cold Drinks & Ice Boost (AI insight)** — weather/event lifts with quick actions (*Order*, *Run 2-for promo*, *Print tag*).
- **Grocery/Retail:**
    - **Categories at Risk (next 48h)** — rows show *Expected sales (48h)*, *Stock you can sell*, *Hours left*, with **Inventory-based/Velocity-based** badge; actions: *Order*, *Substitute*, *Hide*.
- **Liquor:**
    - **Weekend Run-up (Fri–Sun)** — Beer/Wine/Spirits units.

    - **Bundle Suggestions** — AI pairings with *Print shelf tag*.
    - **Top 12 to Re-Order** — *Order = max(0, forecast − on-hand) + Add to PO*.
- **Restaurant:**
    - **Spoilage Risk (next 48h)** — flags items likely to go bad soon based on *prep date*, *storage type*, and *shelf-life*. Columns: *Item*, *Prep date*, *Shelf-life (hrs)*, *Stock you can sell*, *Hours left*, *Risk (High/Med/Low)*. Actions: *Prioritize sell*, *Discount*, *Donate/Discard* → logged to **Waste Ledger**.
    - **Prep Now (Breakfast / Lunch / Dinner)** — suggests quantities to prep now by daypart from forecasted demand and par levels. Actions: *Print prep list*, *Defer to later*.
    - **Running Low** — items projected to run out before next delivery; mirrors 48h risk logic (Expected sales, Stock you can sell, Hours left). Actions: *Order*, *Substitute*.
    - **Waste Ledger** — quick entry log: *Item*, *Qty*, *Reason* (expired, spoilage, mistake), optional *Photo/Note*. Aggregates to a backstage waste% metric and powers future shelf-life tuning.

6.4 Top Items Table

- Shows forecast $ & units per item; user can toggle **Promo?** or tweak **Price**. Clicking **Apply** recomputes lifts client-side and updates KPIs/charts instantly.

6.5 AI Insights

- 2–3 bullets (weather, promos, anomalies). Mirror chart movements. Link to Ask Sunny for deeper Q&A.

6.6 Responsiveness & Accessibility

- **Responsive layouts:**
    - Desktop (≥1280px): 12-column grid; side-by-side charts.
    - Laptop/Tablets (768–1279px): 6–8 columns; charts stack; widgets collapse to cards.
    - Mobile (<768px): single column; KPIs become tiles; tables use horizontal scroll with sticky headers.
- **Browsers:** Latest **Chrome** and **Safari** (desktop & iOS). Degrade gracefully on Firefox/Edge.
- **Touch targets:** 40px min.

- **Contrast:** AA minimum; large text ≥ 16px.
- **Keyboard:** focus styles and tab order defined.
- **Loading & empty states:** skeletons, “No data” copy.

7. Architecture

7.1 Data Ingestion & Feature Store

Sources: POS sales (line-item), prices, promo texts, inventory snapshots, POs/ETAs, weather API, holiday calendar, local events, optional foot-traffic.

Cleaning: dedupe receipts, normalize SKUs, remove impossible values (negative prices), mark **stockout days**.

Feature tables:

- `ts_sales(store_id, sku_id, dt, units, revenue_pre_discounts)`
- `drivers_daily(store_id, dt, promo_lift, weather_lift, holiday_lift, notes)`
- `inventory_snapshot(store_id, sku_id, on_hand, usable, last_count_dt, in_transit_eta)`
- `sku_meta(sku_id, dept, category, perishability, pack_size)`

7.2 Forecasting (LightGBM+ Fallbacks)

- **LightGBM** per store×SKU for **units** and **revenue_pre_discounts**. Regressors: promo size/flag, temperature/heat index, holiday/event dummies.
- Mask stockout days from training.
- **Eligibility:** if ≥180 days history & ≥60% non-zero → LightGBM; else **DMA** (decayed moving average) and **category share-down** for new items.
- Persist **horizon forecasts** (e.g., 35 days) + confidence bands.

7.3 AI Layer (GPT-4 mini)

- Parse promo/event text to structured fields (type, %off, start/end, affected SKUs).
- Compose deterministic **lifts** with learned sensitivities + guardrails (±30%/day/SKU).

- Generate human-readable **insights** and **chart annotations**.
- Label coverage as **Velocity-based** when inventory is stale/unreliable.

7.4 Schedulers

- **Nightly (2am local):** build drivers → train/refresh LightGBM & fallbacks → compose lifts → write `forecast_daily`, `forecast_aggregates`, `insights_daily`.
- **Hourly (every 30–60m):** refresh inventory/promos/weather → recompute *Fast-Mover*, *48h risk*, and liquor *Weekend*.

7.5 APIs (RLS/tenant-safe)

- `GET /api/forecast?stores=&dept=&horizon=&explain=` → KPIs, series (rev/units/actuals), callouts[].
- `GET /api/widgets?type=fastmovers|risk|liquor&stores=&dept=` → widget payloads.
- `POST /api/scenario` { sku_id, promo_flag, price_change } → deltas for KPIs & series (no retrain).
- `POST /api/po` { lines:[{sku_id, qty}] } → PO id/status.

Caching on GETs by (tenant, stores, dept, horizon). Row-level security on all queries.

8. Alerting & Notifications (Demand Forecasting) – v1

Goal

Give store owners an early "heads-up" when demand is about to jump so they can prep staff and stock, without spamming them or auto-changing anything.

- **No auto-orders. No auto-price changes.**
- Alerts are **read-only suggestions** based on DF outputs.

8.1 In-Scope (v1)

1. **Daily email alert** (per tenant / store group) that summarizes:

    a. Upcoming **demand spikes** in the next 3–7 days.
    b. **Today is busier than usual** hints.
    c. Top **categories/items** that are driving the spike (e.g. cold drinks, snacks, ice).
2. **High-urgency push notification** to Modisoft mPOS app when:
    a. **Today** or **next 6 hours** are significantly busier than normal.
3. **Simple rules-based engine** that runs on top of existing DF outputs:
    a. Uses **sales forecasts, holiday flags, and weather-driven lifts** that DF already calculates.
    b. No new ML models for alerts.
4. **Internal alert log / API**
    a. Store alerts in an `alerts_df` table and expose:
        i. `GET /df/alerts?store_id=&date=` so DF dashboard and Sunny can show "Today's alerts".

8.2 Out-of-Scope (v1)

- No **auto-creating purchase orders**.
- No **per-item configurable thresholds** in UI (thresholds are global per business type).
- No multi-channel escalation logic (SMS, voice calls, etc.).
- No LLM-generated emails (body is templated; we only fill numbers and a few reason phrases).

8.3 Alert Types & Rules

All thresholds can live in config so we can tune without code changes.

A. Upcoming demand spike (3–7 days)

Purpose: Warn user that a specific period will be busier than usual so they can plan stock and staffing.

Rule (per store):

- Compute baseline:
  `baseline = typical_avg_sales(store, same_dow_range)`
  (e.g. average of last 4 comparable weekends or weekdays).
- Compute forecast window (e.g. Fri–Sun):
  `forecast_window = sum(forecast_demand[window])`
- If:

```
(forecast_window - baseline) / baseline >= SPIKE_LIFT_PCT
```

- Default `SPIKE_LIFT_PCT`:
  - Convenience: **+25%**
  - Grocery: **+20%**
  - Liquor: **+30%**
  - Restaurant: **+15%**

→ Create a **"Demand Spike"** alert with:

- `window_start`, `window_end`
- `liftPct`
- Top 3 categories by forecast lift.

B. Today busier than usual (same-day)

Purpose: "Today is going to be crazy, get ready now."

Rule (per store):

- Compare **today's forecast** vs typical same weekday:

```
today_lift = (forecast_today - typical_weekday) / typical_weekday
```

- If `today_lift >= TODAY_LIFT_PCT` (default **+20%**):

→ Create **"Today Busy"** alert with `today_lift` and top 2–3 categories.

This alert can drive both the daily email and **same-day mPOS push**.

C. Weather / holiday driven category spikes

Purpose: Highlight specific categories that will move more due to known events.

Rule (per store × category):

- Use DF's feature outputs/feature store (weather & holiday impact). If not exposed, approximate via:

```
cat_lift_pct = (forecast_cat_next_3d - baseline_cat_next_3d) /
baseline_cat_next_3d
```

- If `cat_lift_pct >= CAT_LIFT_PCT` (default **+15%**) **and** category ∈ {Cold Drinks, Ice, Beer, Snacks, etc.}:

→ Add this category under "Categories with biggest jump".

8.4 Frequency & Channels

1. **Daily batch job (per store timezone)**
- Run once per day after the DF run completes (e.g. **06:00 local time**).
- Generate alerts for the **next 3–7 days**.
- Aggregate into **one email per tenant** (or per store group, depending on config).
2. **Same-day push notifications (mPOS)**
- Lightweight job every **60 minutes** during open hours:
  - Re-evaluate Today Busy rule.
  - If it crosses the threshold and **no push has been sent today**:
    - Send **one push notification** to mPOS for that store.
  - No more than **1 DF push per store per day**.

8.5 Data Inputs

The alert engine **reuses existing DF outputs**:

- Per store × day:

- `forecast_units`, `forecast_revenue`
- `baseline_units`/`baseline_revenue` or approximate from history.
- Calendar flags: `is_holiday`, payday, `event_flag`.
- Weather drivers: `temp_bucket`, `rain_flag`, etc.

No need for external model APIs. All work is inside Modisoft's DF pipeline.

8.6 Backend Design (DF Alerts Service)

Scheduled job (Prefect / cron / Azure job):

1. Fetch forecast & baseline for `{store, date}` over horizon `today .. today+7`.
2. Compute:
   a. `today_lift_pct`
   b. `spike_windows` (e.g. Fri–Sun, next weekend).
   c. Category-level lifts for key categories.
3. For each store, build an **alert payload JSON**:

```
{
  "storeId": "c101",
  "date": "2025-11-06",
  "df": {
    "todayBusy": { "liftPct": 0.24 },
    "spikes": [
      { "windowStart": "2025-11-08", "windowEnd": "2025-11-10",
"liftPct": 0.32 }
    ],
    "categories": [
      { "name": "Cold Drinks", "liftPct": 0.35, "window": "Fri-Sun" },
      { "name": "Ice", "liftPct": 0.28, "window": "Fri-Sun" }
    ]
  }
}
```

4. Persist to `alerts_df` table:
- `tenant_id, store_id, alert_date, type, severity, payload_json, created_at, channels_sent.`

5. Send:

- **Email** → call internal email service with aggregated payload.
- **Push** → call mPOS push API for `todayBusy` alerts only.

API:

- `GET /df/alerts?store_id=c101&date=2025-11-06`
    - Returns latest alerts payload (so DF dashboard & Sunny can show it).

8.7 Email Template (DF)

Subject options

- `Heads up: demand is jumping this weekend at {StoreName}`
- `This week’s forecast for {StoreName} (drinks, snacks, ice)`

Body skeleton

Hi {FirstName},

Here’s a quick look at the next few days for **{StoreName}**.

1. Busier than usual

- **{Day / Date Range}:** about **{liftPct}% higher** than a normal {weekday/weekend}.
- Main drivers: **{“warm weather / holiday / event”}**.

2. Categories with the biggest jump

- **Cold drinks: ~{units} units** expected
- **Snacks: ~{units} units**
- **Ice: ~{units} bags**

What to do

- Make sure shelves are full before **{key day/time}**.
- Move an extra cooler or display to the front if possible.

These numbers come from your recent sales plus weather/holiday data. They don't change your inventory or orders automatically.

(Backend just fills `{StoreName}`, `{liftPct}`, `{units}`, `{reason}` etc.)

8.8 Push Notification Text (DF)

Examples (one per store per day max):

- `Today will be busier than usual at {StoreName}. Check cold drinks & snacks.`
- `Heatwave coming Fri–Sun. Cold drinks forecast is up ~30%. Plan extra stock.`

Tap action: deep link into DF dashboard (or Alerts panel) for that store.

8.9 Acceptance Criteria (DF Alerts)

1. **Correct triggers**
    a. When forecast window is ≥ configured **SPIKE_LIFT_PCT**, a "Demand Spike" alert appears in `alerts_df` and email.
    b. When today's forecast is ≥ **TODAY_LIFT_PCT**, a "Today Busy" alert appears and push notification is sent once.
2. **No spam**
    a. At most **one** DF email per tenant per day.
    b. At most **one** DF push per store per day.
3. **Data correctness**
    a. Lift % values in email match calculations from forecast vs baseline (tolerance ±1%).
    b. Category units in email equal DF category forecasts (± rounding).
4. **Sunny / dashboard parity**
    a. `GET /df/alerts` returns the same alerts that were emailed (same lift %, same categories).
5. **Performance**

a. Daily DF alert job runs within **5 minutes** for all stores after DF pipeline finishes.
b. `GET /df/alerts` p95 latency ≤ **500 ms**.

9. Detailed Logic

9.1 Lifts

- Weather: heat index deltas map to % by dept (e.g., beverages +8–12%).
- Promos: price elasticity tables by category; cap extremes; combine multiplicatively with weather/holiday.
- Events/holidays: additive dummies with priors; AI fills gaps (maps text→category).

9.2 Widgets

- **Fast-Mover Refill:** `expected_6h = (today_units_forecast/12)*6; refill_now = max(0, expected_6h - on_shelf)`.
- **48h Category Risk:** `expected_48h = sum(next 48h units)`; `usable = on_hand - unsellable (+ in_transit if ETA ≤48h)`; `hours_left = usable / (expected_48h/48)`; badge rule = inventory snapshot freshness & sanity.
- **Liquor Weekend:** sum Fri–Sun units by Beer/Wine/Spirits from `final_forecast`.

9.3 Scenario Engine (client-first)

- Apply promo/price edits locally: `final = base × (1 + lift)`; update KPIs/charts immediately.
- Persist edit via `POST /api/scenario` for audit; nightly pipeline incorporates confirmed promos.

10. Non-Functional Requirements

- **Performance:** P95 dashboard load <2s on 14-day range; API P95 <600ms; Lighthouse perf ≥ 85 on desktop and ≥ 75 on mobile.

- **Reliability:** Jobs idempotent; 3 retries with backoff; alert if >30m late.
- **Security:** Tenant isolation (RLS), HTTPS, audit logs on scenario/PO actions.
- **Privacy:** No PII in training; data retention 13 months (configurable).
- **Cost:** AI batch budget < $0.02/store/day; Pro chat gated behind user click.
- **Internationalization (phase 2):** currency symbol/format; timezones per store.

11. Acceptance Criteria

1. All 6 KPIs render with tooltips and correct values for selected filters.
2. Charts show Actual vs Forecast; Explain overlays appear on affected days.
3. Convenience fast-mover table computes **Refill now** correctly and prints a list.
4. Grocery 48h risk list shows **Inventory-based** when `last_count_dt ≤24h` & sane; else **Velocity-based**; actions work.
5. Liquor widgets: Weekend Run-up, Bundle Suggestions, Top-12 Re-Order with **Add to PO**; `Order = max(0, forecast – on_hand)`.
6. Scenario edits (promo/price) update KPIs & charts without reload in <600ms.
7. Ask Sunny link contains filters + KPIs; Sunny reply references those numbers.
8. **Responsive:**
    a. Mobile: single-column, horizontal scroll for tables, sticky headers; font ≥16px.
    b. Tablet: two-column; charts stack; controls collapse.
    c. Desktop: 12-column; side-by-side charts.
    d. Verified on **Chrome** (Win/Mac/Android) and **Safari** (iOS/macOS).
9. Data Health increments for: negative on_hand; stale count >7d; missing sku_meta; broken SKU mapping.

12. Analytics & Observability

- Usage: DAU/WAU, time on page, scenario edits, clicks on *Order/Substitute/Refill/Add to PO*, Sunny clicks.
- Forecast quality (internal): MAPE/WAPE by store/SKU; drift detection.
- Pipeline health: job success/latency; last success timestamp; alerting to Slack/Pager.

- AI cost: tokens/day per task; budget alarms.

13. Rollout Plan

1. Pilot 4–6 stores per vertical for 2 weeks; collect stock-out %, waste %, and owner feedback.
2. Tune lift tables & category mappings; address data health.
3. General availability with Sunny enabled by default.

14. Risks & Mitigations

- **Dirty inventory:** velocity badge + prompts to count; exclude extreme negatives.
- **Promo mis-parsing:** rule fallbacks + manual override; cache parsed promos.
- **Over-responsive weather:** per-store coefficients + lift caps.
- **Cost creep:** cache, short prompts, batch minis only; Pro usage behind click.
- **Timezones:** run nightly per store timezone to avoid misalignment.

15. Glossary

- **Revenue (before discounts):** sales without coupons/markdowns.
- **Lift:** % boost/cut from promo/weather/events.
- **Velocity-based coverage:** hours left computed from recent sell rate when inventory is unreliable.
- **Coverage (hours left):** usable stock ÷ current hourly sales.
- **DMA:** decayed moving average.

16. Change Log

- **v1.1** — Added detailed backend flow, responsiveness & browser support, acceptance tests, widget logic, cost/observability; clarified business-type behavior.
- **v1.2** - Changed ML model from Prophet to LightGBM and added Alerting & Notifications (Section 8) to give store owners an early “heads-up” when demand is about to jump so they can prep staff and stock, without spamming them or auto-changing anything