# NUMEN MVP MASTER ROADMAP

## Purpose

This document is the execution plan for the next build sequence of Numen/Nutrition Autopilot. It is written for direct use by Claude Code inside the repo.

This is **not** a greenfield build. This is a structured continuation of an existing working system. Build incrementally. Preserve the current pipeline. Prioritize scientific rigor and operational usability.

---

## Operating Mode

### Full-Run Mode

When asked to "run full," execute all sprints in sequence **without re-planning the whole system each time**.

For each sprint:

1. Inspect current extension points only (schema/API/UI/tests impacted by that sprint)
2. Implement smallest safe version of required scope
3. Add/extend tests before UI polish
4. Run validations + regressions
5. Update handoff docs
6. Gate pass/fail
7. Continue to next sprint only if gate passes

If a gate fails:

- stop advancing to the next sprint
- stabilize current sprint
- document blocker and exact next steps

### Build Style

- Engineering-first
- Incremental
- No architecture recap unless needed for a decision

- No broad rewrites of stable modules
- No fake certainty
- No hidden state changes

---

## Global Non-Negotiables

### Scientific Rigor

- Preserve deterministic nutrient math behavior
- Preserve provenance tracking for nutrient values
- Preserve immutable label freeze semantics
- Preserve auditability (traceable inputs, weights, lots, nutrient sources)
- No silent substitutions or silent assumptions
- Document assumptions in `ASSUMPTIONS.md`

### Pipeline Safety

Do not break:

- SOT import
- Instacart import
- enrichment pipeline
- schedule flow
- Fed/Skip serving flow
- immutable label freeze
- label generation

### MVP Scope Discipline

This roadmap is for:

- 1 client
- 1 small kitchen
- PWA-first chef/admin usage

Do not build:

- multi-kitchen orchestration
- complex roles/permissions
- native mobile priority work (Expo remains secondary)
- speculative autonomous planner behavior that bypasses review

---

## Required Handoff Files (update every sprint)

- `PROGRESS.md`
- `OPEN_ISSUES.md`
- `SCIENTIFIC_RISKS.md`
- `MORNING_HANDOFF.md`

Each sprint must update these with:

- shipped work
- remaining work
- test status
- changed files summary
- migration notes (if any)
- exact commands to run locally
- next sprint recommendation

---

## Validation Gate (must pass before advancing)

A sprint is considered gated/passed only if all are true:

- tests green (including new sprint tests)
- typecheck green
- lint green
- web build green (if UI touched)
- no regression in import → enrichment → schedule → label freeze
- `SCIENTIFIC_RISKS.md` updated with any new risk/assumption
- migration/apply status documented if schema changed

If any fail, fix or clearly document blocker before continuing.

---

## Rollback / Safety Rules

- Work on a feature branch unless explicitly told otherwise
- Commit in logical checkpoints
- Do not squash away scientific test history during active sprinting
- If a migration is risky, prefer additive schema changes over destructive changes

- Never mutate historical label snapshots in debug/recompute tools

---

## Sprint Sequence (Dependency-Ordered)

---

# Sprint 1 — Inventory Intelligence + Instacart Mapping UX + Substitution Engine (MVP)

## Objective

Improve operational reliability and speed in kitchen/admin workflows by adding:

1. inventory intelligence (projections / allocation / reorder signals)
2. fast Instacart mapping review UX
3. practical substitution engine with scientific rigor + auditability

## Scope (Required)

### A. Inventory Intelligence

Build:

- projected availability by date/time window using:
  - inventory lots
  - meal schedules
  - batch prep demand
  - allocations/reservations
- 7-day inventory demand forecast:
  - by ingredient/component family
  - required vs available vs shortfall
- par levels / reorder thresholds (MVP-simple, configurable)
- allocation visibility:
  - on-hand / allocated / available / projected balance
- waste analytics (MVP-light):
  - quantities + counts by reason and ingredient/component

UX:

- surface high-signal states:
    - shortage this week
    - critical today
    - overallocated
    - expiring soon
- use clear badges/status colors (do not bury risk in tables)

## B. Instacart Mapping UX

Build:

- unmapped line item queue (sortable/filterable)
- candidate match suggestions ranked by confidence:
    - UPC exact
    - normalized name similarity
    - brand match
    - size/unit similarity
    - historical accepted mappings
- one-click actions:
    - approve suggested match
    - search/select different existing SKU/ingredient
    - create new SKU + link to canonical ingredient
    - mark pantry/non-tracked with reason
- mapping memory / learned mappings
- dry-run + apply workflow

Requirements:

- preserve import idempotency / duplicate protections
- preserve provenance/confidence metadata
- record mapping resolution source (manual vs auto-suggest accepted)

## C. Substitution Engine (MVP)

Build substitution suggestions for planned meals/batches using:

- same component family
- allergen/exclusion compatibility
- inventory availability
- prep readiness / batch status
- nutrient delta score
- optional flavor compatibility tags (if available)

Add:

- explainable ranking ("why suggested")
- preview before apply (before/after nutrient delta)
- apply action for future planned meals/batches only
- audit event/reason logging

Constraints:

- no silent substitutions
- no changes to served/frozen labels
- warnings when uncertainty is high

### D. Scientific QA + Regression Hardening

Add tests for:

- inventory allocation invariants
- projection logic correctness
- mapping memory + idempotency preservation
- substitution constraints/ranking/deltas
- deterministic nutrient recalculation post-substitution
- rounding + calorie sanity after substitution
- no impact on served/frozen labels

Extend QA/admin exceptions to show:

- unresolved mapping queue counts
- overallocated inventory
- substitution warnings
- projected shortages affecting scheduled meals

## Sprint 1 Acceptance Criteria

- Instacart mapping review queue is usable end-to-end
- inventory page surfaces projected shortages and overallocation
- substitutions can be previewed and applied to future plans only
- scientific and regression tests pass
- no label freeze regressions

---

# Sprint 2 — Batch Yield Calibration + Cook/Chill QC Telemetry (MVP)

## Objective

Replace static yield assumptions with measured production data and capture QC checkpoints that improve scientific rigor and planning accuracy.

## Scope (Required)

### A. Batch Yield Calibration

Build:

- yield profile history (expected %, actual %, variance, sample count, confidence)
- calibration dimensions (MVP):
    - component/batch type
    - method
    - optional cut/form factor
- suggested yield update workflow (human accept required)
- outlier handling (flag + exclude from calibration proposal)

### B. Cook/Chill QC Telemetry

Build:

- checkpoint logging:
    - cook start/end
    - target vs actual temp
    - chill start/end
    - chill compliance
    - hold status/notes
- step requirements before state transitions (MVP-simple)
- issue flags:
    - temp miss
    - chill time exceeded
    - missing/late checkpoint
    - manual override with reason

### C. Batch Variance Analytics

Build:

- expected vs actual yield summaries
- trend over time
- repeated variance flags
- top problematic batches/components
- light contributor views (method / notes / patterns if available)

### D. Planning Integration

Planning/projection logic should:

- prefer approved calibrated yields
- fall back to defaults if confidence insufficient
- explicitly show basis used (default vs calibrated)

No silent replacement of assumptions.

## Scientific QA + Tests

Add tests for:

- calibration proposal logic
- outlier rejection
- calibrated/default selection
- checkpoint gating invariants
- QC override requirements
- deterministic planning math with calibrated yields
- no impact on frozen labels
- calorie/rounding sanity remains green

Extend QA/admin exceptions to show:

- missing required checkpoints
- repeated yield variance issues
- unreviewed calibration proposals
- QC overrides requiring review

## Sprint 2 Acceptance Criteria

- checkpoint data can be logged in kitchen flow
- calibrated yields are proposed and reviewable
- planning can use calibrated yields with explicit traceability
- QA surfaces QC misses and calibration risks

- tests/regressions green

---

# Sprint 3 — Menu Composer + Weekly Prep Optimizer + Sauce Matrix (MVP)

## Objective

Operationalize composition-first planning:

- standardized components (protein/base/veg/sauce)
- minimized prep complexity
- personalized flavor via sauces

## Scope (Required)

### A. Menu Composer

Build a composition-first planner:

- choose protein/base/veg/sauce
- set gram targets / presets
- macro/calorie preview
- save reusable composition template
- compatibility guidance (tags)
- allergen/incompatibility warnings
- flavor swaps via sauce variants without duplicating core prep assumptions

### B. Weekly Prep Optimizer (MVP heuristic)

Build:

- 7-day component demand rollup
- cooked quantity required
- raw quantity estimate via approved/default yields
- bundling recommendations (share neutral batches across multiple meals)
- prep draft generation (review before commit)
- shortage/blocker surfacing integrated with inventory intelligence

### C. Sauce Matrix

Build:

- sauce matrix UI:
    - flavor family
    - macro variant
    - compatibility tags
- default sauce assignments for component combinations/templates
- portion presets (5g/10g/15g/20g...) with macro delta preview
- flavor rotation support (MVP-light) to reduce repetition

### D. Schedule Integration (coexist recipe-first + composition-first)

Build compatibility path so planned meals can use composition templates without breaking existing recipe-based schedule flow.

Requirements:

- preserve label freeze and nutrient traceability
- preserve QA visibility
- if compatibility layer is needed, implement and document clearly

## Scientific QA + Tests

Add tests for:

- composition macro aggregation determinism
- sauce portion/variant effects
- yield-adjusted prep rollup calculations
- inventory shortage detection in planning
- compatibility/allergen warning logic
- recipe-first + composition-first coexistence behavior
- no regression in label freeze pipeline

Extend QA/admin warnings for:

- composition templates with estimated nutrient inputs
- rollups using low-confidence yields
- shortages blocking prep draft generation

## Sprint 3 Acceptance Criteria

- composition templates can be created and previewed
- weekly prep rollup produces usable batch suggestions
- sauce matrix supports practical personalization

- existing schedule/label flow still works
- tests/regressions green

---

# Sprint 4 — Client Biometrics Timeline + DEXA/Bloodwork/CGM Ingestion Layer (MVP)

## Objective

Create a structured client data layer for time-series biometrics and document ingestion with auditability and future parsing hooks.

This sprint is storage/indexing/timeline-first. Do not fake parsers.

## Scope (Required)

### A. Client Biometrics Timeline

Build:

- biometric snapshots (date-stamped)
  - height
  - weight
  - body fat % (optional)
  - lean mass (optional)
  - notes/source
- timeline UI:
  - chronological records
  - latest values summary
  - add/edit snapshots
  - missing/irregular data indicators
- trend preview (MVP-light):
  - up/down/stable indicators

### B. Document Ingestion Layer

Build structured client document records:

- type (`DEXA`, `BLOODWORK`, `CGM`, `OTHER`)
- date collected
- date uploaded
- source/provider (optional)
- tags
- file linkage/storage metadata
- parsing status (`not_started`, `queued`, `parsed_partial`, `verified`, `failed`)

Add upload/management UI:

- upload file
- assign type/date
- view metadata
- mark verified
- add notes
- filter/search by type/date/status

### C. Parsed Metrics Scaffold (future-ready, MVP-safe)

Create normalized metric series storage:

- metric key
- value + unit
- observed_at
- source document reference
- confidence/verification status

Add manual entry path for common metrics (MVP):

- fasting glucose
- LDL/HDL/triglycerides
- HbA1c
- body fat % / lean mass
- optional resting HR

No fake parsing. Keep explicit manual/placeholder workflow when parser is absent.

### D. QA/Admin Integration

Surface:

- missing recent biometrics
- unverified documents
- failed parsing attempts (if any)
- stale metric data affecting planning confidence (MVP rule-based)

## Scientific QA + Tests

Add tests for:

- time-series ordering and latest selection
- document metadata integrity
- metric linkage to source documents
- parsing status transitions
- verification flags
- no impact on existing meal/label pipeline

No silent unit assumptions; keep source + verification states explicit.

## Sprint 4 Acceptance Criteria

- client timeline usable
- document upload/metadata workflow usable
- metrics can be stored with provenance/verification
- QA warns on stale/unverified data
- tests/regressions green

---

# Sprint 5 — Audit Trace + Reproducibility + Ops Control Tower (MVP)

## Objective

Make the system operationally and scientifically legible:

1. audit trace visibility
2. reproducibility/debug tooling
3. high-signal admin control tower

## Scope (Required)

### A. Label/Meal Audit Trace Viewer

For a selected served meal/label snapshot, show:

- schedule/service event context
- freeze-time recipe/composition inputs
- ingredients/components + gram weights
- inventory lots consumed
- nutrient provenance for contributing values
- calculation summary (human-readable)
- QA warnings/confidence flags present at freeze time

Requirements:

- preserve immutable snapshot semantics
- distinguish stored snapshot vs current data if both are shown
- no historical mutation

## B. Reproducibility / Debug Tools (non-destructive)

Build:

- recompute preview diff (historical snapshot vs current computed result)
- delta explanation (mapping changes, nutrient source changes, yields, etc.)
- freeze-time integrity checks (references present and internally consistent)

No mutation of snapshots. Diff-only.

## C. Ops Control Tower Dashboard

Build a single high-signal dashboard with sections:

- Today:
  - meals due
  - batches due/active/blocked
  - shortages/overallocations
  - expiring inventory
- Scientific QA:
  - unresolved verification tasks
  - estimated/inferred nutrient counts
  - missing provenance/incomplete coverage
  - substitution warnings pending review
  - calibration/QC override review items
- Client Data Readiness:
  - stale biometrics/docs
  - unverified uploads
- System Reliability (MVP-light):
  - failed imports
  - failed enrichment jobs

    - stuck statuses
- Attention Queue:
    - top actionable issues with direct links

### D. Exportable Review Packs (MVP)

Print/export-friendly views:

- Daily Ops Summary
- Scientific QA Summary
- Meal Audit Summary (single meal/label)

Browser print is acceptable. Prioritize correctness and readability.

## Scientific QA + Tests

Add tests for:

- audit trace integrity and provenance display logic
- recompute preview diff behavior (non-destructive)
- dashboard aggregation correctness
- attention queue deterministic prioritization (if scoring used)
- no mutation of label snapshots during debug paths
- regression coverage for current pipeline

## Sprint 5 Acceptance Criteria

- audit trace viewer explains freeze-time label generation
- recompute diff is safe and useful
- control tower surfaces highest-risk issues clearly
- export/print review views are usable
- tests/regressions green

---

## Full-Run Execution Protocol (Claude Code)

## Start Sequence

1. Read this file.
2. Detect current codebase state and which sprint is next.
3. If no sprint-tracking marker exists, start at Sprint 1.

4. Create/update a sprint tracking file:
    - `ROADMAP_STATUS.md`
    - mark current sprint, status, gate result, notes
5. Execute current sprint.
6. Run gate.
7. If gate passes, advance `ROADMAP_STATUS.md` and continue.
8. If gate fails, stop advancement and document blockers precisely.

## Sprint Tracking Format (use this exact shape)

Maintain `ROADMAP_STATUS.md` with:

- Current sprint
- Status (`not_started`, `in_progress`, `gated_pass`, `blocked`)
- Last completed gate timestamp
- Blockers
- Next action

## Implementation Discipline

For each sprint:

- inspect extension points only
- implement data model/API changes first (where needed)
- add tests for invariants before UI polish
- implement UI
- run validations/regressions
- update docs
- commit checkpoint(s)

## Testing / Validation Commands

Discover and use repo-native commands from `package.json` / workspace scripts. Document exact commands in `MORNING_HANDOFF.md`.
Do not invent command names. Verify before writing docs.

---

## Stop/Ship Criteria for MVP Funding Demo

The roadmap can be considered demo-ready when all are true:

- Sprints 1–5 gated pass
- core pipeline remains stable
- chef/admin PWA workflows are usable in real kitchen simulation
- scientific QA and audit trace are visible and trustworthy
- control tower surfaces operational and scientific risk clearly
- handoff docs are current and credible

---

## Default Prioritization Rule (when time/complexity tradeoffs appear)

Choose in this order:

1. Scientific correctness / auditability
2. Pipeline safety
3. Chef/admin operational usability
4. Speed/automation convenience
5. UI polish

If a tradeoff is made, document it in `SCIENTIFIC_RISKS.md` or `OPEN_ISSUES.md`.

---

## Final Instruction to Claude

Build directly against current context and current repo state. Do not reset. Do not genericize. Ship the highest-leverage increment that preserves scientific rigor and operational momentum.