Food Department Management App — Project Summary

Last Updated: February 7, 2026
Architecture: Single-file HTML application
Tech Stack: React 18, Dexie (IndexedDB), Tailwind CSS, Babel (in-browser transpilation)

App Overview

A comprehensive food department management app designed to handle the complete workflow from ingredient management through dish preparation. The app tracks availability, quantities, priorities, and spoilage across ingredients, intermediates (preparations), and dishes — with an auto-generated shopping list.

Completed Phases

Phase 1 — Core Foundation (`index.html`)

- **Ingredients tab:** CRUD operations, stock tracking, spoilage engine (Fresh/NearExpiry/Expired badges)
- **Dishes tab:** CRUD with recipe ingredient linking, priority (1–5), status (Planned/InProgress/Cooked)
- **Availability engine:** Per-dish evaluation — checks if all required ingredients have sufficient stock
- **Cook workflow:** Deducts ingredient stock, sets dish status to Cooked
- **Buy workflow:** Increments ingredient stock, updates `purchasedAt` timestamp
- **Database:** Dexie v2 with `ingredients` and `dishes` stores
- **Design:** Warm kitchen palette — terracotta (`#C4704F`), sage (`#87A878`), cream (`#FDF8F3`)

Phase 2 — (Incorporated into Phase 3)

Phase 3 — Shopping List (`index-phase3.html`)

- **Shopping List tab:** Auto-generated from active (non-Cooked) dishes
- **Quantity Aggregation Engine:** Sums ingredient requirements across all active dishes
- **Deficit calculation:** `deficit = max(0, totalRequired - stockQty)`
- **Priority Propagation Engine:** Ingredients inherit the highest priority (lowest number) from their dependent dishes
- **Shopping list filters:** "Need to Buy" vs "All Items"
- **Buy from shopping list:** Pre-fills suggested quantity based on deficit
- **Priority badges:** Urgent/High/Normal/Low/Someday with color coding

- **Badge on nav:** Shows count of items to buy

Phase 4 — Intermediates System (`index_R004_phase4_app.html`)

- **Preps tab:** New tab with plum/purple (`#6C5CE7`) theming for managing intermediates (doughs, sauces, broths)
- **Intermediate entity:** Has name, unit, stockQty, and `inputIngredients` array (each with ingredientId and qtyPerUnit)
- **Prepare workflow:** Deducts input ingredients, increments intermediate stock
- **Dish → Intermediate linking:** Dishes can require both ingredients AND intermediates
- **Updated Availability Engine:** Checks both ingredient and intermediate sufficiency for dishes
- **Updated Quantity Aggregation:** Expands intermediate deficits into ingredient requirements for shopping list
- **Updated Priority Propagation:** Flows through intermediates to their input ingredients
- **Database:** Dexie v3 with added `intermediates` store
- **4-tab navigation:** Ingredients → Preps → Dishes → Shop

Phase 4 Bug Fix

- Fixed PrepDlg: Added unit labels ("Quantity (g)"), helper text, ingredient stock preview after preparation, insufficient stock validation with disabled Prepare button
- Fixed BuyDlg: Added unit labels to Quantity field and Now/After display
- Fixed Cook dialog: Added units to deduction lines
- Fixed Shopping list: Added units to Have/Need/Buy values

Current App Architecture

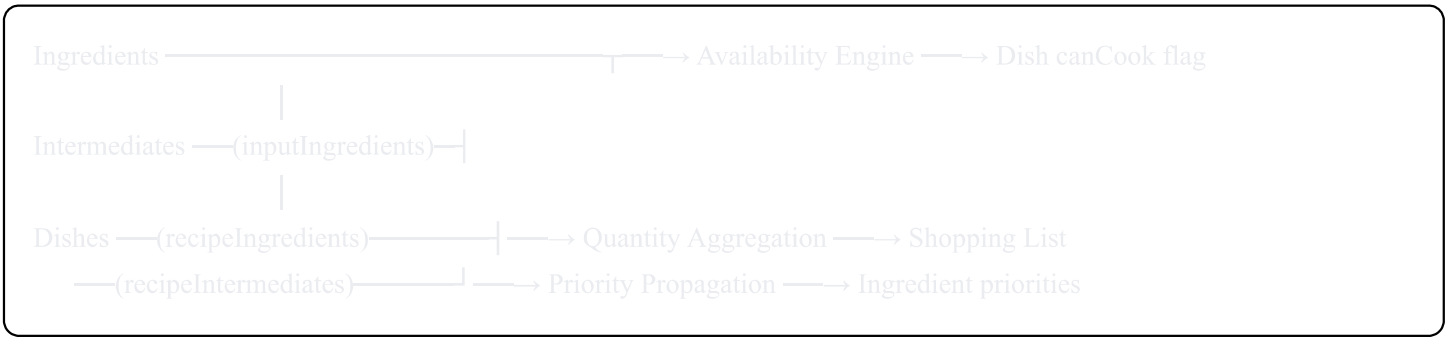
Database Schema (Dexie v3)

```
ingredients: id, name, unit, stockQty, shelfLifeDays, purchasedAt
intermediates: id, name, unit, stockQty, inputIngredients[]
  └─ inputIngredients[]: { ingredientId, qtyPerUnit }
dishes: id, name, status, priority, recipeIngredients[], recipeIntermediates[]
  └─ recipeIngredients[]: { ingredientId, qty }
  └─ recipeIntermediates[]: { intermediateId, qty }
```

Core Engines

1. **Spoilage Engine:** Computes expiryDate, daysRemaining, spoilageStatus from purchasedAt + shelfLifeDays
2. **Availability Engine:** Per-dish check — are all ingredients AND intermediates in sufficient stock?
3. **Quantity Aggregation Engine:** Cross-dish summing of ingredient/intermediate needs, deficit calculation, intermediate-to-ingredient expansion for shopping list
4. **Priority Propagation Engine:** Ingredients inherit highest priority from dependent dishes (via both direct use and intermediate chains)

Data Flow



Pending Phases (Planned)

Phase 5 — Data Integrity & Workflow Guards

- **Atomic transactions:** Wrap Cook/Prepare workflows in `db.transaction()` for rollback capability
- **Validation guards:** Block cooking unavailable dishes, block preparing with insufficient inputs, block zero-qty purchases, block re-cooking cooked dishes, block buying non-existent ingredients
- **Negative stock prevention:** Clamp or reject operations that would cause negative values
- **File management approach:** Use `str_replace` (section-level edits) instead of full file regeneration to manage growing file size

Phase 6 — Import/Export

- **JSON export:** Full database dump (all 3 entity types)
- **JSON import with validation:** Required field checks, reference integrity (dish → ingredient, dish → intermediate, intermediate → ingredient, price → shop), negative stock rejection, invalid status rejection
- **Overwrite mode:** Confirmation-gated full replacement
- **Atomic import:** Entire import succeeds or rolls back — no partial imports

Phase 7 — Ingredient Monitoring & Smart Alerts

Three distinct monitoring strategies per ingredient:

Strategy 1 — Expiry-driven (vegetables, meat, dairy):

- Ingredients that spoil fast
- Alert: "Spinach expires in 2 days"
- Suggestion: "You have Palak Paneer and Spinach Dal that use spinach — cook one?"
- Goal: Use-it-or-lose-it notifications with recipe suggestions

Strategy 2 — Usage-driven (eggs, butter, oil):

- Ingredients consumed gradually outside tracked recipes
- Feature: Manual deduction ("used 50g") + low stock threshold
- Alert: "Butter is below 100g, running low"
- Goal: Track consumption that doesn't go through the Cook workflow

Strategy 3 — Interval-checked (rice, sauces, long-shelf staples):

- Expiry is months away, portion tracking impractical
- Feature: Periodic reminders ("It's been 2 weeks — check your rice stock")
- User manually confirms/updates quantity
- Goal: Catch depletion of hard-to-track staples

Shared design for all three:

- Each ingredient gets a configurable **monitoring mode** field
- Alerts appear in a dedicated **Alerts/Suggestions** view
- Nothing auto-adds to shopping list — all alerts require **user confirmation** before becoming purchase requests
- Contextual intelligence: e.g., if you're going on a trip, no point buying bulk items

---

Test Scenarios (52 Total)

Category	Count	Coverage
Availability	8	Unlinked dishes, zero stock, partial missing, cooked exclusion, intermediate checks

Category	Count	Coverage
Quantity Aggregation	8	Shared ingredients, shared intermediates, direct+indirect overlap, partial expansion, no-active-dish edge case
Priority Propagation	6	Min-priority inheritance, cooked exclusion, intermediate flow-through, no-dependency edge case
Spoilage & Expiry	6	Missing timestamps, near-expiry threshold, expired-but-in-stock, partial usage
Workflow Actions	10	Cook/prepare blocking, buy validation, atomic deductions, re-cook prevention, multi-input preparation
Import & Sync	8	Duplicate names, missing fields, broken references, overwrite, invalid status/stock
Storage Atomicity	6	Mid-operation crash rollback, concurrent action serialization, post-restart recomputation

---

Known UX Issues (To Address)

Mobile Input Friction

- Number input fields don't auto-select on focus — typing appends to default value (e.g., `0` → `01000`)
- **Fix:** Add `onFocus={e => e.target.select()}` to all number inputs in future phase

File Size Management

- Single-file HTML is approaching generation limits
- Phase 4 code uses compressed variable names to fit
- **Discussed options:** Section-level str_replace edits (Option C recommended), multi-file split, build step
- **Decision:** On hold — to be revisited before next major phase

---

Design System

Element	Color	Hex	Usage
Terracotta	Primary	#C4704F	Buttons, active nav, ingredients
Sage	Success	#87A878	Cook actions, in-stock indicators
Purple/Plum	Preparations	#6C5CE7	Preps tab, intermediate theming

Element	Color	Hex	Usage
Tomato	Danger/Missing	#E55039	Expired, missing items, deficits
Cream	Background	#FDF8F3	Page background
Charcoal	Text	#2D3436	Primary text
Warm Gray	Secondary text	#636E72	Labels, descriptions
Butter	Warning	#F6E58D	Near-expiry badges

File Inventory

File	Phase	Description
index.html	Phase 1	Core foundation — ingredients + dishes
index-phase3.html	Phase 3	Added shopping list, aggregation, priority propagation
index_R004_phase4_app.html	Phase 4	Added intermediates system, 4-tab layout (latest)