Scranly — Database Strategy Plan (MVP)

1. Objectives

Design a relational schema that is easy to evolve, enforces data integrity, and performs well for the MVP use■cases.

Keep the model human■readable without sacrificing normalization and query efficiency.

2. Scope & Assumptions (MVP)

- Users (accounts + auth linkage)
- Recipes (core metadata, nutrition, media)
- Plans (weekly) and Plan Meals (day/slot → recipe)
- Shopping Lists and List Items (derived + manual additions)
- Basic audit fields (created_at, updated_at)
- Soft delete via deleted_at where helpful

3. Entity Model (ERD Outline)

Core: users, recipes, recipe_nutrition, recipe_media, plans, plan_meals, shopping_lists, shopping_list_items.

Reference lookups: meal_slot (breakfast, lunch, dinner, snack), unit (count, g, ml), tag (optional).

Optional (defer if time): pantry_items, recipe_ingredients, user_preferences.

4. Table Design (Draft)

- users(id PK, email UNIQUE, display_name, auth_provider, created_at, updated_at)
- recipes(id PK, title, desc, time_minutes, calories, protein_g, carbs_g, fat_g, image_url, author_user_id FK users.id NULLABLE, created_at, updated_at)
- plans(id PK, user_id FK users.id, week_start DATE, created_at, updated_at, UNIQUE(user_id, week_start))
- plan_meals(id PK, plan_id FK plans.id, day DATE, slot meal_slot, recipe_id FK recipes.id, note TEXT NULL, created_at, updated_at, UNIQUE(plan_id, day, slot))
- shopping_lists(id PK, user_id FK users.id, week_start DATE, created_at, updated_at, UNIQUE(user_id, week_start))
- shopping_list_items(id PK, list_id FK shopping_lists.id, name, aisle, need_amount NUMERIC(10,2), need_unit unit, emoji TEXT NULL, is_checked BOOL DEFAULT FALSE, est_price_per_pack NUMERIC(10,2) NULL, pack_amount NUMERIC(10,2) NULL, pack_unit unit NULL, size_label TEXT NULL, created_at, updated_at)
- meal_slot(slot TEXT PK CHECK slot IN ('breakfast','lunch','dinner','snack'))
- unit(code TEXT PK CHECK code IN ('count','grams','milliliters'))

5. Naming & Conventions

- snake_case table and column names

- Singular columns, plural tables
- id as PK (UUID v7 preferred), created_at/updated_at TIMESTAMPTZ NOT NULL
- FKs named __fkey
- ENUM■like lookup tables (meal_slot, unit) or real SQL ENUM if you prefer strictness

6. Integrity & Constraints

- NOT NULL on all required fields
- CHECK constraints for numeric ranges (e.g., time_minutes >= 0)
- UNIQUE(user_id, week_start) to prevent duplicate plans/lists
- FK ON DELETE CASCADE for plan_meals → plans; RESTRICT where data loss is risky
- Use generated columns only if they replace repeated computation cleanly

7. Indexing Strategy

- PK indexes (implicit) on all id columns
- Foreign key indexes: plan_meals(plan_id), plan_meals(recipe_id), shopping_lists(user_id), shopping_list_items(list_id)
- Week queries: plans(user_id, week_start), shopping_lists(user_id, week_start)
- Searchable text (name/title) → simple btree with prefix ops now; consider trigram/GIN later

8. Partitioning & Archival (Later)

Not needed for MVP scale. Consider week_start RANGE partitions for plans/shopping_lists if weekly data becomes large (>50M rows).

Set up an archival policy for stale lists (> 1 year) into cold storage if costs rise.

9. Security & Access

- DB roles: app_readonly, app_rw, migration_admin
- Row Level Security (optional for MVP): enable per table; policies user_id = current_app_user()
- Never return soft■deleted rows by default (deleted_at IS NULL)

10. Migrations & Seeding

- Adopt a migration tool (Flyway, Goose, Prisma, or SQLx migrations) and commit SQL
- One baseline migration, then small, reversible steps
- Seed fixtures: one user, 5 recipes, 1 weekly plan, 5 plan_meals, 1 shopping_list + 15 items

11. Performance Checklist (Local)

- Top 5 queries: EXPLAIN ANALYZE < 80 ms on laptop DB
- All FKs have supporting indexes
- No seq scans on hot paths (unless table < 1k rows)
- Pagination uses keyset or limit/offset with stable ORDER BY

12. Observability & Backups

- Enable slow query log (> 250 ms) in staging/prod
- Daily full backup + WAL archiving (if PostgreSQL)
- Add basic table/column COMMENTs for ERD autodocs

13. Change Management

- Backwards■compatible changes first (additive)
- Dual■write or backfill scripts for breaking changes
- Feature flags or versioned views when removing columns

14. Definitions of Done

- Migrations apply cleanly on empty DB
- Seed script runs and app boots with demo data
- Schema comments added for non■obvious columns
- EXPLAIN on key queries looks healthy

15. Checklist (Copy into your repo)

- [ ] Create baseline migration
- [ ] Add lookup tables (meal_slot, unit)
- [ ] Add users, recipes, plans, plan_meals
- [ ] Add shopping_lists, shopping_list_items
- [ ] Seed ‘golden week’ fixtures
- [ ] Index pass + EXPLAIN
- [ ] Add table/column comments
- [ ] Backup policy noted in README