Scranly — MVP Architecture, DDD Overview & Roa

Generated 2025-10-23 11:21 UTC

This document summarises the MVP scope, a domain-driven design (DDD) outline, C4 diagra (Context & Containers), a pragmatic schema for the operational store, an append-only events store, API surface for the iOS app, and a phased delivery plan.

- Keep the MVP small and focused; iterate with real usage data.
- Model behaviours in domains; persist facts as events where useful.
- Separate operational reads from analytics/insights workloads.

MVP Scope (v1)

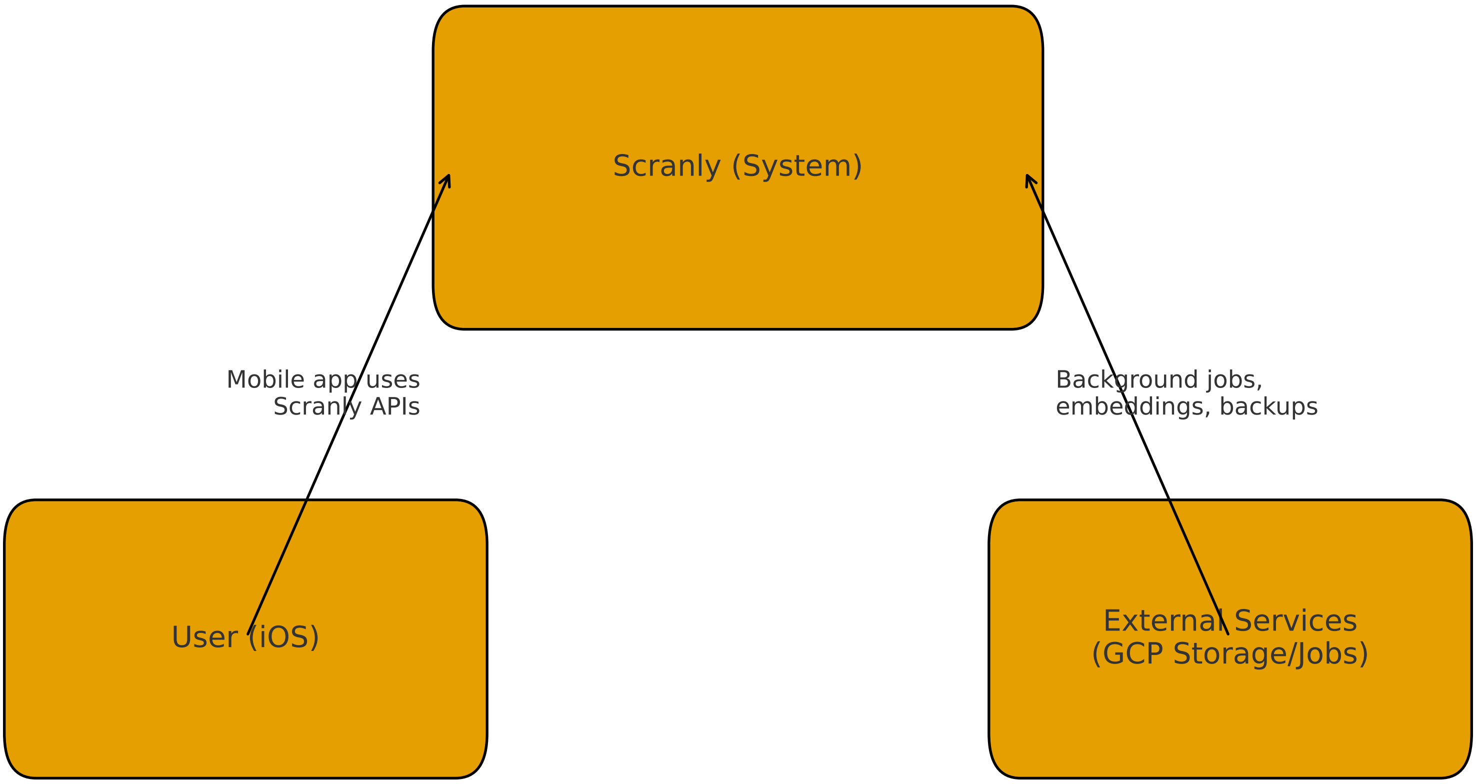
- Home: stat carousel, quick fun-fact/quote, and “next meal” spotlight.
- Discover: Daily Bites (3 picks/day) and Scranly Chef Insights (weekly teaser), plus Netflix-style carousels with Cook/Bin actions, and a Liked list.
- Plan: dinner-only for MVP; sticky week header, bottom day carousel, quote + ‘up next’ card, and a compact ‘time saved’ bar.
- Shop: grocery list by aisle for the latest plan, stats/education card, export list; manual prices for now.
- Track: today’s calories ring, simple macros bars, 7-day calories bars with goal line; goals editable.

DDD Bounded Contexts

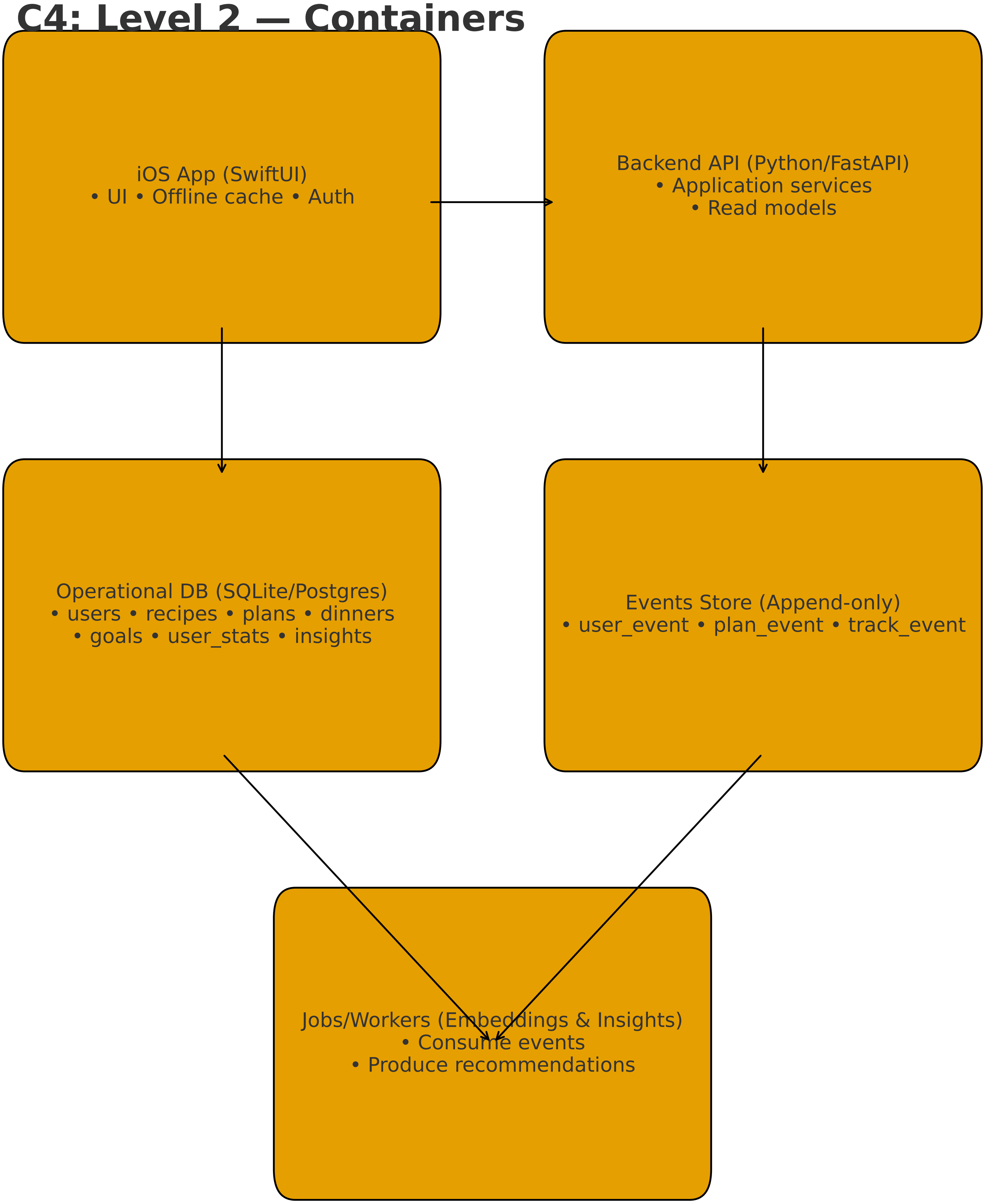
- Identity & Profiles — users, preferences, personalisation score.
- Discovery — recommendations, Daily Bites selection, Chef Insights generation.
- Planning — weekly plan, dinner slots only (MVP), plan adjustments.
- Shopping — grocery list generation, aisle grouping, export.
- Tracking — nutrition goals, daily totals, rolling 7-day visualisations.

Guiding principles: rich domain objects encapsulate behaviour; use events to record decisions (liked/removed/cooked/goal-updated). Build thin read models for SwiftUI needs.

C4: Level 1 — System Context



C4: Level 2 — Containers



iOS App (SwiftUI)
• UI • Offline cache • Auth

Backend API (Python/FastAPI)
• Application services
• Read models

Operational DB (SQLite/Postgres)
• users • recipes • plans • dinners
• goals • user_stats • insights

Events Store (Append-only)
• user_event • plan_event • track_event

Jobs/Workers (Embeddings & Insights)
• Consume events
• Produce recommendations

Table	Columns
users	user_id (PK), email, created_at, display_name
recipes	recipe_id (PK), title, desc, image_url, calories, protein_g, carbs_g, fat_g,

Stream	Structure / Examples
user_event	event_id (PK), user_id, event_type, occurred_at, payload_json — e.g., RecipeLiked, RecipeBinned, PersonalisationScoreUpdated

Endpoint	Description
GET /v1/discover/daily-bites	Return 3 recipes for today (title, kcal, time, image_url, ids).
GET /v1/discover/insights/weekly	Return teaser payload (copy + link token) for Chef Insights.

Phased Delivery Plan

- Phase 0 — Scope & Cuts: lock MVP features, dinner-only planning, drop basket price estimate.
- Phase 1 — UI: Home, Discover (Daily Bites + Insights teaser + carousels), Plan (week header, day carousel), Shop (list + export), Track (calories ring + simple macros + 7-day bars).
- Phase 2 — Data & API: operational schemas, CRUD endpoints, small seed set of recipes, user_stats with personalisation_score.
- Phase 3 — Behaviour & Insights: append-only events; embeddings/worker for recommendations; Ch Insights generator.
- Phase 4 — Polish & Beta: app polish, empty states, analytics, TestFlight; iterate on Daily Bites quality.

Rule of thumb: ship narrow, instrument well, and re-rank with real feedback.