Phase 1 Implementation Plan – Meal Planning Application (Verbatim)

Below is a detailed, execution-ready Phase 1 implementation plan derived directly from the reviewed architecture and phased build document you provided.

This plan is written at the level expected for an internal engineering kickoff: concrete deliverables, sequencing, ownership boundaries, and explicit non-goals.

Phase 1 Objective

Establish a deterministic, production-grade foundation for the meal planning application by implementing core data models, CRUD workflows, recipe seeding, vector indexing, and baseline UI screens. Phase 1 intentionally avoids cooking logic, feasibility checks, grocery lists, or LLM orchestration beyond data preparation.

Success criteria:
- All core entities persisted in MongoDB with clean schemas and indexes
- Recipes are seeded, embedded, and searchable via Qdrant
- Users can manage pantry items and recipes end-to-end via the UI
- System is ready for Phase 2 business logic without schema refactors

Phase 1 Scope (Explicit)

In Scope:
- Pantry inventory CRUD (ingredient + packaged items)
- Recipe CRUD and deterministic storage
- Idempotent recipe seeding from recipes_seed.json
- Vector embedding + Qdrant indexing
- Foundational React UI
- API contracts and error handling

Out of Scope:
- Unit conversion logic
- Cooking feasibility checks
- Grocery list generation
- Meal planning logic
- LLM / MCP runtime interaction

1. Data Model & Persistence Layer (MongoDB)

1.1 Core Collections

PantryLots
Unified representation for ingredients and packaged items.

Design notes:
- No normalization between ingredient vs packaged at Phase 1
- Quantity may reach zero but is never auto-deleted
- No unit canonicalization yet (user units preserved)

Indexes:
- { name: 1 }
- { type: 1 }

Recipes
Deterministic recipe storage with ingredients, instructions, servings, tags.

Indexes:
- { name: 1 }
- { tags: 1 }

1.2 Repository Layer (Spring Data Mongo)

Deliverables:
- PantryLotRepository
- RecipeRepository
- Explicit indexes declared via annotations or migration script

Non-Goals:
- No aggregation pipelines
- No feasibility queries

2. Spring Boot Backend (System of Record)

2.1 REST API Design

Pantry APIs:
- GET /api/pantry
- POST /api/pantry
- PUT /api/pantry/{id}
- DELETE /api/pantry/{id} (manual only)

Recipe APIs:
- GET /api/recipes
- GET /api/recipes/{id}
- POST /api/recipes
- PUT /api/recipes/{id}
- DELETE /api/recipes/{id}

Design rules:
- All validation enforced server-side
- Clear 4xx vs 5xx error semantics
- No side effects across domains

2.2 Recipe Seeding

Deliverables:
- Startup or CLI-triggered seeder
- Reads recipes_seed.json
- Idempotent by recipe name or external ID
- Safe to re-run in all environments

Example behavior:
- Existing recipe → skipped or updated (explicit strategy chosen)
- New recipe → inserted and queued for embedding

3. Vector Pipeline (Python + Qdrant)

3.1 Embedding Job

Deliverables:
- Python batch script:
- Loads recipes from Mongo
- Generates embeddings (local model)
- Writes to Qdrant

Embedding input (deterministic):
Recipe Name
Ingredients list
High-level instructions

3.2 Qdrant Collection

Collection config:
- Vector size = embedding dimension
- Distance = cosine
- Payload:
- recipeId
- recipeName

Optional (recommended):
- Store vectorId back in Mongo recipe document

Non-Goals:
- No live semantic search in UI yet
- No hybrid ranking

4. Frontend (React + TypeScript + Tailwind)

4.1 Application Shell

Deliverables:
- Routing scaffold
- API client abstraction
- Global error handling
- Basic layout (responsive, web-first)

4.2 Pantry Screens
- Pantry list view
- Add / edit pantry item
- Quantity editing
- Manual delete action

4.3 Recipe Screens
- Recipe list
- Recipe detail view
- Recipe editor (ingredients + instructions)
- Servings editable (no scaling logic yet)

5. Cross-Cutting Concerns

5.1 Validation
- Required fields enforced in backend
- UI performs optimistic validation only

5.2 Error Handling
- Structured error responses from backend
- UI displays actionable messages

5.3 Configuration
- Environment-specific Mongo and Qdrant configs
- Seed job toggleable per environment

6. Phase 1 Milestones & Sequencing

1. Schemas + Repositories
2. CRUD APIs
3. Recipe Seeding
4. Embedding + Qdrant Indexing
5. Frontend Pantry UI
6. Frontend Recipe UI
7. End-to-End Validation

7. Phase 1 Exit Criteria

Phase 1 is complete when:
- A user can manage pantry and recipes end-to-end
- Recipes are embedded and queryable via Qdrant
- No schema changes are required to begin Phase 2 cooking logic

- All business rules remain deterministic and backend-enforced