useFoods Hook – Developer Guide

A React hook for listing, adding, and searching Foods stored in the on-device SQLite database via Drizzle ORM (Expo).

Purpose

• Encapsulates DB access so screens/components don’t write SQL.
• Normalizes user input (strings → integers, trimmed names).
• Prevents duplicate names by default (configurable).
• Exposes a simple API: items, loading, error, refresh, addFood, searchFoods.

Prerequisites & Assumptions

• App is wrapped with SQLiteProvider and your DB is initialized/migrated before screens render.
• You have useDrizzle() available (built from useSQLiteContext()).
• Schema matches: name, calories, carbs, proteins, fats, createdAt.

API

- items: Food[] – current foods in memory (newest first).
- loading: boolean – true while fetching/adding.
- error: Error | null – last load/add error (if any).
- refresh(): Promise – re-fetches the list.
- addFood(input: NewFoodInput, options?): Promise – inserts a food record.
- searchFoods(term: string): Promise – LIKE search by name.

Behavior Details

• Initial load: Automatically fetches foods ordered by createdAt DESC.
• Normalization: trims name, converts numeric strings to integers.
• Duplicates: prevented by default unless allowDuplicateName=true.
• After insert: refresh() is called to reload items.
• Return: addFood returns the most recently added or existing row.
• Search: case-insensitive LIKE match by name.

Common Usage

const { items, loading, error, addFood, refresh } = useFoods(); await addFood({ name: "Oats", calories: 150, carbs: 27, proteins: 5, fats: 3 }); const matches = await searchFoods("oat"); await refresh();

Error Handling & Testing

• Wrap addFood in try/catch to surface validation or DB errors.
• Use error + refresh for reload UI.
• Unit test with mocked useDrizzle(), integration test with temp DB file.

Performance Tips

• Batch inserts before refresh() for speed.
• For large lists, paginate (LIMIT/OFFSET).
• Add index on foods(name) for faster search.