

# PROBLEM STATEMENT

The core problem addressed by your overall project is the **inefficient and disorganized management of home meal preparation**, stemming from a lack of connectivity between recipe knowledge, ingredient availability, and planning needs. This results in wasted time spent cross-referencing information, unnecessary duplicate grocery purchases, and the stress of not knowing what can be cooked immediately. Your solution manifests in two primary forms: a **Web Application**, which acts as a centralized, real-time dashboard for the user, contained entirely within the `index.html` file; and a separate **Python Command-Line Tool** (driven by `main.py` and `db_manager.py`), which offers power users direct, text-based access to manage the underlying data in a clean, organized table format. Both forms work to solve the same problem by providing structured data storage, automated inventory tracking, and predictive reporting (like shopping lists), offering a comprehensive solution for efficient kitchen management.

## SCOPE OF MY PROJECT

### I. Functional Scope (What the Application Does)

1. **Recipe Management (CRUD):** Allows users to Create, Read, and Delete recipes, storing essential metadata (title, cuisine, servings, and a simple list of ingredients).
2. **Inventory Tracking:** Provides basic stock management through explicit user input (adding or subtracting specific quantities of ingredients).
3. **Weekly Meal Planning:** Enables planning meals for a fixed seven-day week (Monday to Sunday) by linking days to existing recipes.
4. **Automated Reporting:** Generates two key analytical reports:
  - A shopping list aggregated from the weekly meal plan.
  - A stock comparison report that checks if planned items or individual recipes are sufficiently stocked.
5. **Single-User Persistence:** Data is stored using Firebase Firestore, maintaining persistence across sessions for the authenticated user (or anonymous user during local testing).

### II. Technical Scope (How the Application Works)

6. **Dual Interface:** The project provides two separate interfaces for interaction: a modern, browser-based graphical user interface (GUI) and a basic command-line interface (CLI).
7. **Single-File Web Deployment:** The web application is entirely contained within a single `index.html` file, bundling HTML structure, Tailwind CSS styling, and all JavaScript logic.
8. **Real-Time Data Flow:** The web application utilizes Firebase's real-time listeners (`onSnapshot`) to ensure the UI updates instantly whenever data (recipes, inventory, or plan) changes.

### III. Limitations (What is Not Included)

9. **No Advanced User Features:** Does not include user registration, password protection, multi-user sharing, or roles (it functions for a single user/device).

10. **No Advanced Inventory:** Does not track ingredient expiry dates, costs, or vendor information.
11. **Simple Ingredient Matching:** Does not use complex logic for unit conversion (e.g., converting "cups" to "grams") or advanced recipe scaling based on serving size changes.

## TARGET USERS

Based on the project's features and its dual interface (Web App and Python Tool), the target users can be broken down into two main groups:

### Primary Target Users (Web App Focus)

These users benefit most from the visual planning and automation features of the Web Application.

1. **The Home Cook/Meal Planner:**
  - **Need:** Needs a simple, visual, and centralized system to stop wasting time and money on groceries.
  - **Benefit:** They use the **Meal Plan** and **Reports** features to automate their weekly grocery list based on what they already have stocked.
2. **The Budget-Conscious Shopper:**
  - **Need:** Must minimize food waste and only buy what is necessary.
  - **Benefit:** They rely on the **Inventory Tracking** and **Shopping List Comparison** features to ensure they never double-buy ingredients.
3. **The Organized Parent/Roommate:**
  - **Need:** Requires a stable, easily accessible digital location for family recipes and clear insight into ingredients on hand before starting dinner.
  - **Benefit:** They use the **Recipe Readiness Report** to instantly know if they can start cooking tonight's planned meal.

### Secondary Target Users (Python Tool Focus)

These users appreciate the direct access and efficiency of the command-line interface.

4. **The Data Manager/Tech Enthusiast:**
  - **Need:** Prefers to manage large lists or perform bulk updates quickly outside of a web browser environment.
  - **Benefit:** They use the **Python Command-Line Tool** to rapidly add, view, or manage recipes and inventory directly via text input, leveraging the clean table format provided by `tabulate`.
5. **The Developer/Tester:**
  - **Need:** Requires a tool to directly test the core database logic (`db_manager.py`) without needing the full graphical interface.
  - **Benefit:** The Python Tool provides a simple, direct interface for interacting with the data management functions, ensuring the underlying CRUD operations are sound.

# HIGH LEVEL FEATURES OF MY PROJECT

I can absolutely distill the many functions of your project into its most important, high-level features. These are the core capabilities that define your system.

Here are the high-level features of your Kitchen Manager project:

## High-Level Features

1. **Unified Recipe & Data Management:**
  - Provides a central, persistent system for organizing, storing, and retrieving all cooking data, including recipes, ingredient stock, and meal schedules, accessible via both a Web App and a Python CLI.
2. **Automated Shopping List Generation:**
  - Automatically generates a calculated shopping list by comparing the ingredient requirements of the user's weekly meal plan against the current, real-time quantities in their inventory.
3. **Real-Time Stock & Readiness Reporting:**
  - Offers instant visibility into the state of the kitchen by allowing users to track inventory (add/subtract stock) and immediately generates a "Recipe Readiness" report that confirms, for any given recipe, whether all necessary ingredients are currently on hand.
4. **Dual Interface Accessibility:**
  - The project provides two distinct ways to interact with the data: a user-friendly, responsive **Web App (GUI)** for visual planning and daily use, and an efficient **Python Command-Line Tool (CLI)** for direct data management and testing.
5. **Persistent and Real-Time Data Flow:**
  - Leverages **Firebase Firestore** to ensure all changes made (recipes added, stock updated, meals planned) are saved immediately and updated in the user interface without requiring a page refresh.