Title: Diet Manager (YADA) Design Document

Date: 08-04-2025

– Inesh Shukla (2023113008)
– Krish Jalan (2023101074)

---

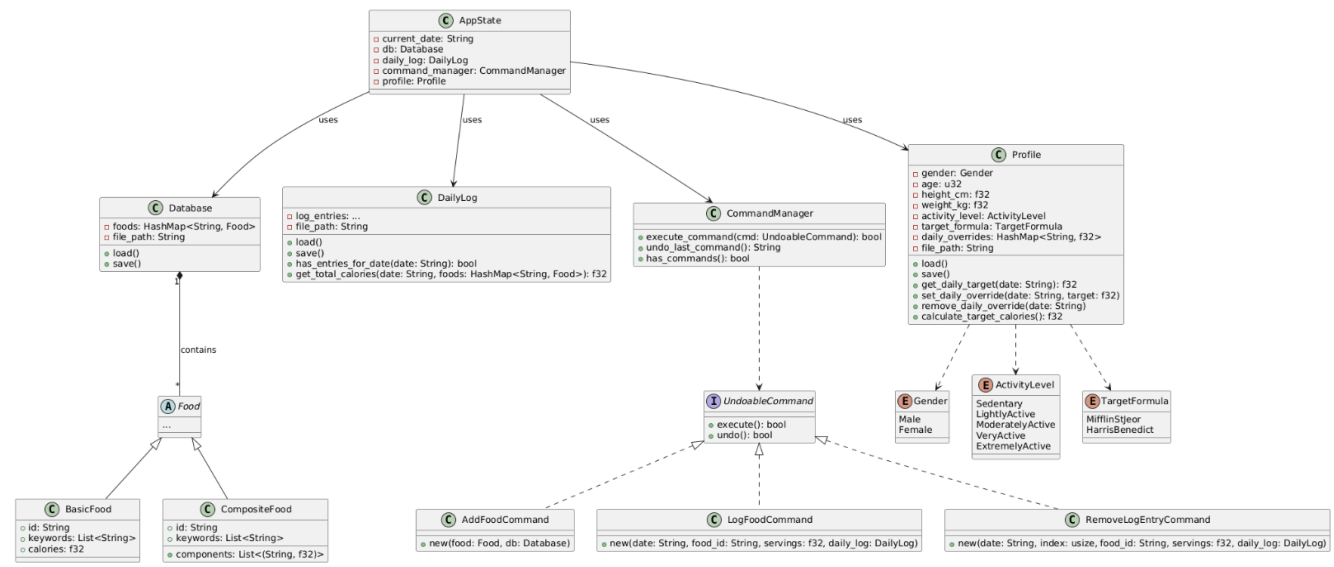
1. Overview

The Diet Manager (YADA) is a command-line application designed to help users manage their dietary data. The product allows users to add and search foods (both basic and composite), log daily food consumption, view nutritional summaries, manage personal profiles, and navigate dates. Key features include:

- Food Database Management (adding basic and composite foods)
- Food Search and Logging (by keyword with multiple matching strategies)
- Daily Log and Summary (calorie counting and food history)
- Profile Management (personal attributes and target calorie setting)
- Undo functionality using command pattern
- Data persistence via JSON files

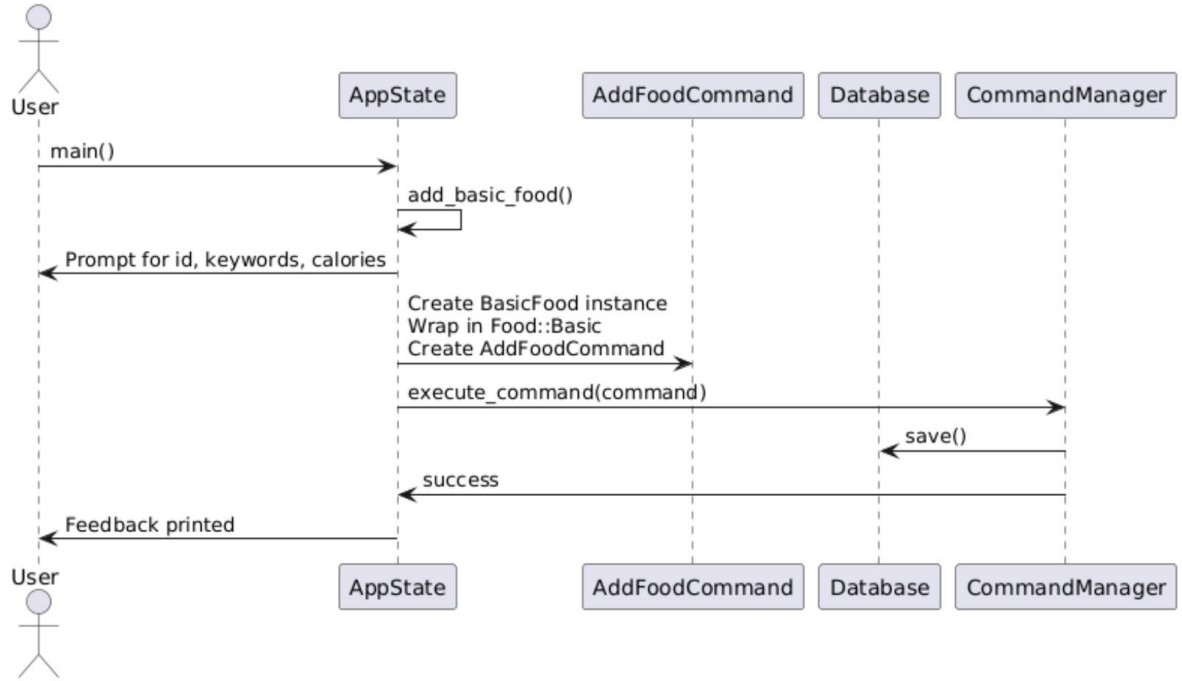
---

2. UML Class Diagram



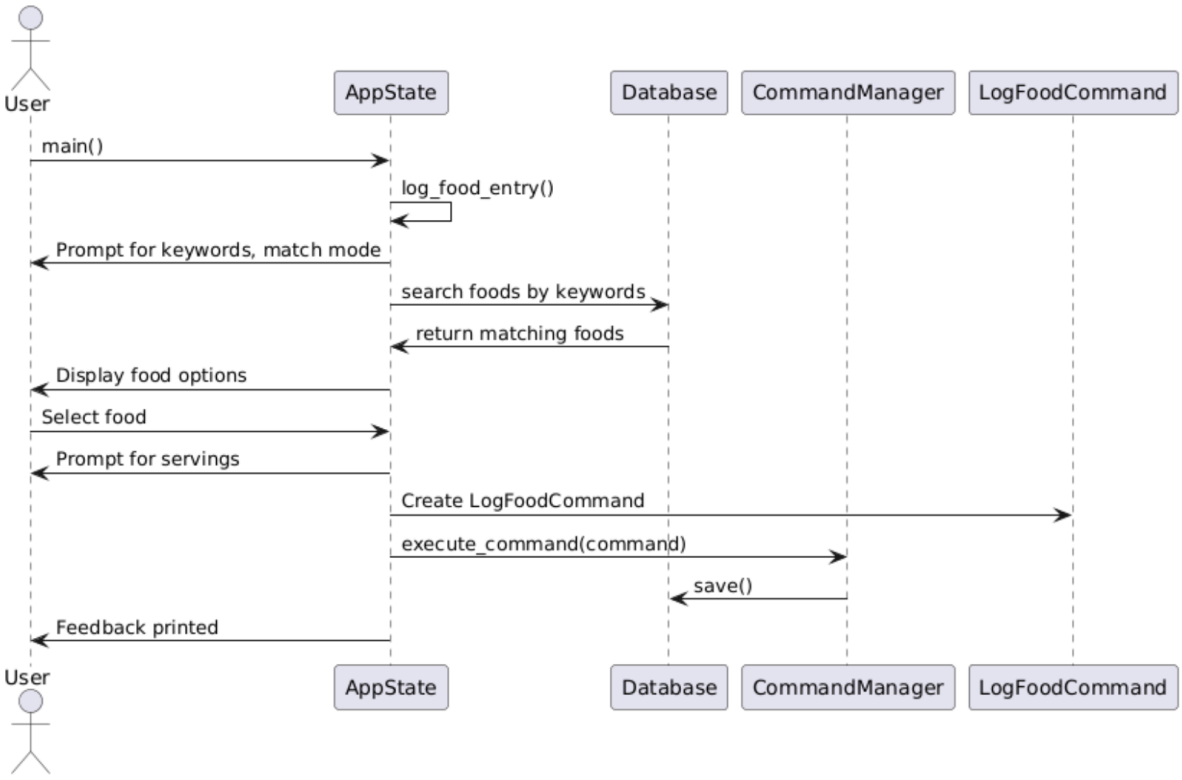
3. Sequence Diagrams

a. Adding a Basic Food



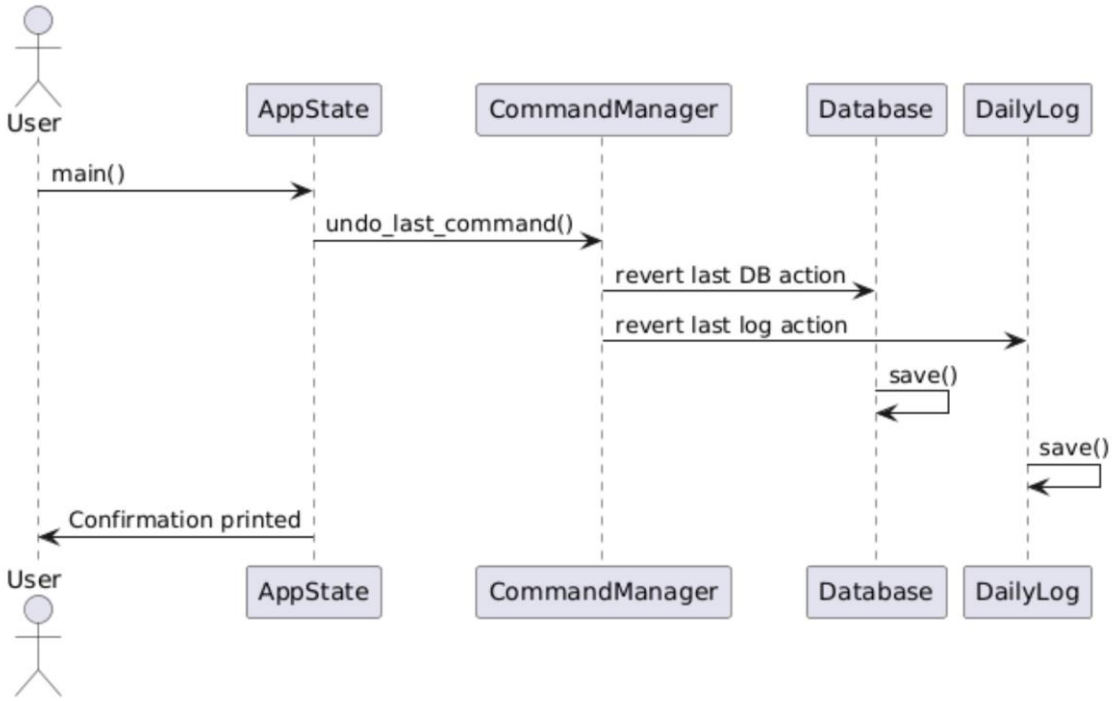
---

b. Logging Food Entry via Keyword Search

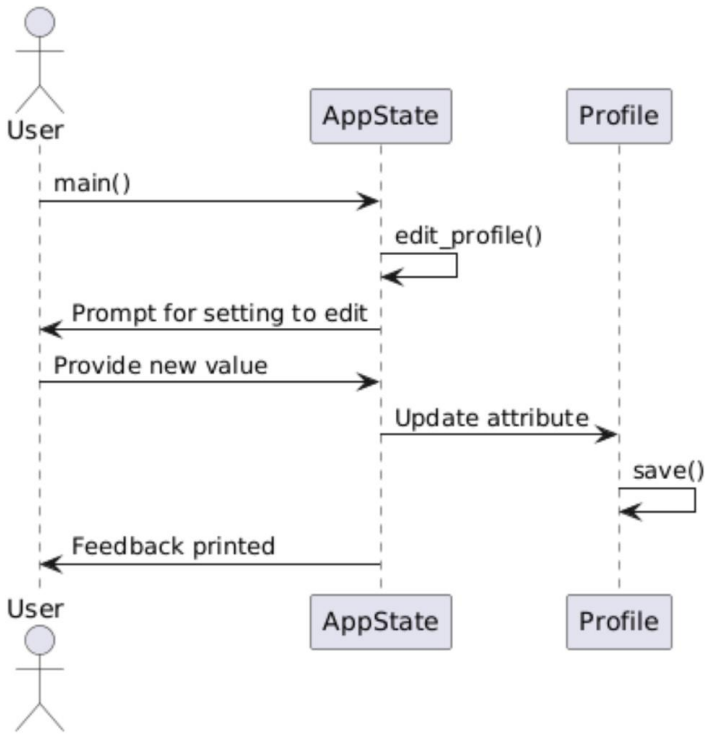


---

c. Undo Last Command



d. Changing Profile Settings



4. Design Narrative

The design reflects a balance among critical software quality attributes:

• Low Coupling and High Cohesion:
– Each module (database, logging, command processing, profile management) encapsulates its own functionality.
– The AppState structure aggregates related modules without them directly depending on one another.

• Separation of Concerns and Information Hiding:
– The Database and DailyLog classes manage file I/O and persistence, keeping persistence logic separate from business logic.
– The Command pattern isolates change operations and undo functionality.

• Adherence to the Law of Demeter:
– Most modules interact via clearly defined interfaces (for example, using methods such as load(), save(), execute_command()).

• Extensibility:
– With the command hierarchy, new operations can be added without modifying existing code extensively.

---

5. Reflection

Strong Aspects:
• Clear separation of concerns – Data persistence, business logic, and command processing are distinct.
• Extensibility through the command pattern – New operations (e.g., additional logging or food operations) can be added easily.

Weak Aspects:
• User interaction tightly coupled with I/O prompts – Could be further abstracted for different interfaces.
• Error handling is minimal (e.g., parsing and file I/O) – Enhancing robustness by better error management is an area for improvement.
