Design Document for YADA (Yet Another Diet Assistant)

Team Members: [Insert Names Here]

March 28, 2025

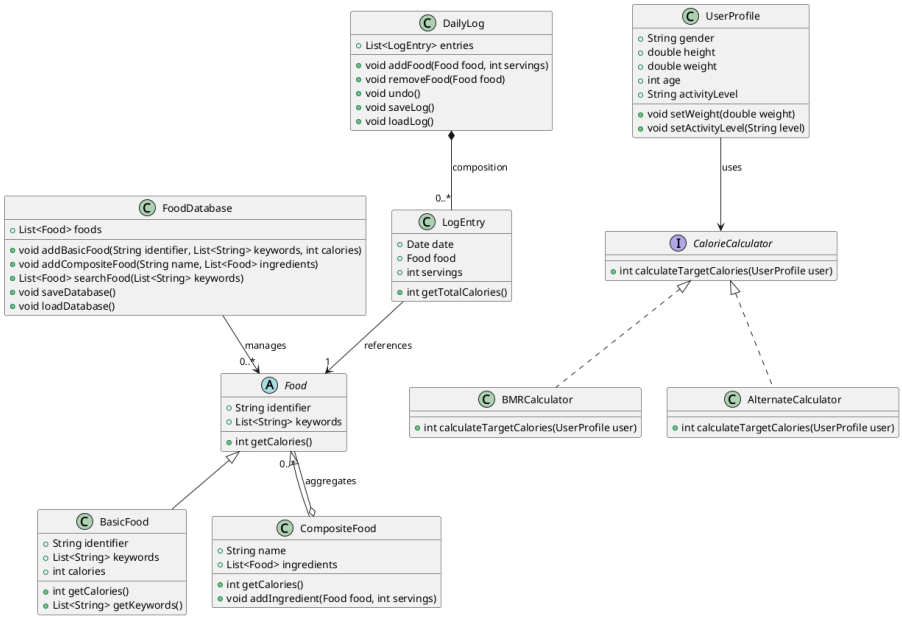
Overview

YADA (Yet Another Diet Assistant) is a software prototype designed to help users manage their diet and calorie intake. The application provides features such as maintaining a food database, logging daily food consumption, and calculating target calorie intake based on user profiles. Key features include:

- A food database supporting both basic and composite foods.
- Daily logs for tracking food consumption.
- User profiles with customizable calorie calculation methods.
- Undo functionality for log modifications.
- Dynamic calorie goal calculation based on user activity and profile.

UML Class Diagram

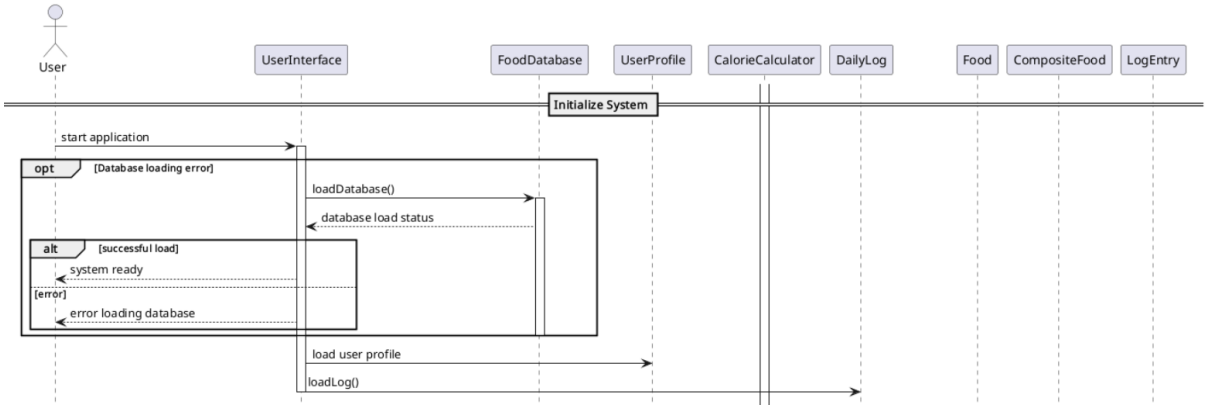
The following UML class diagram illustrates the main classes and interfaces in the design, along with their relationships (inheritance, association, aggregation, and composition). Cardinality and role indicators are included for clarity.



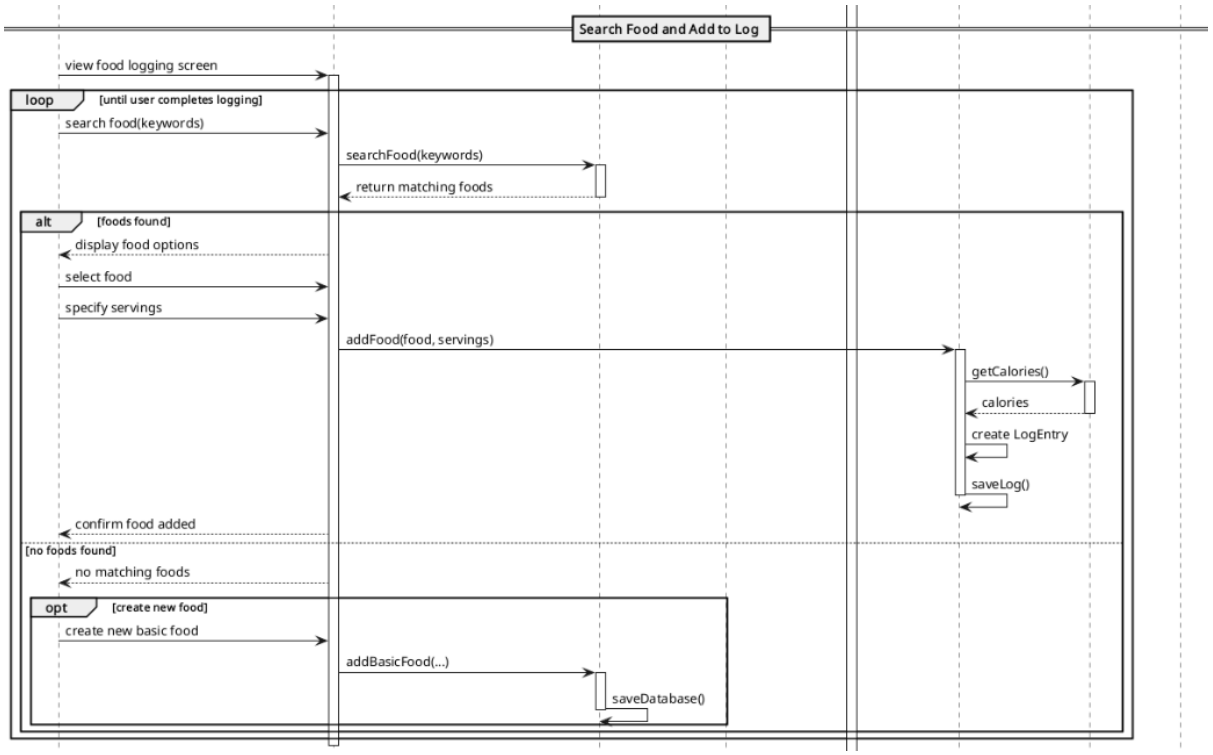
Sequence Diagrams

The following sequence diagrams provide insight into key interactions in the system. Each diagram corresponds to a major use case.

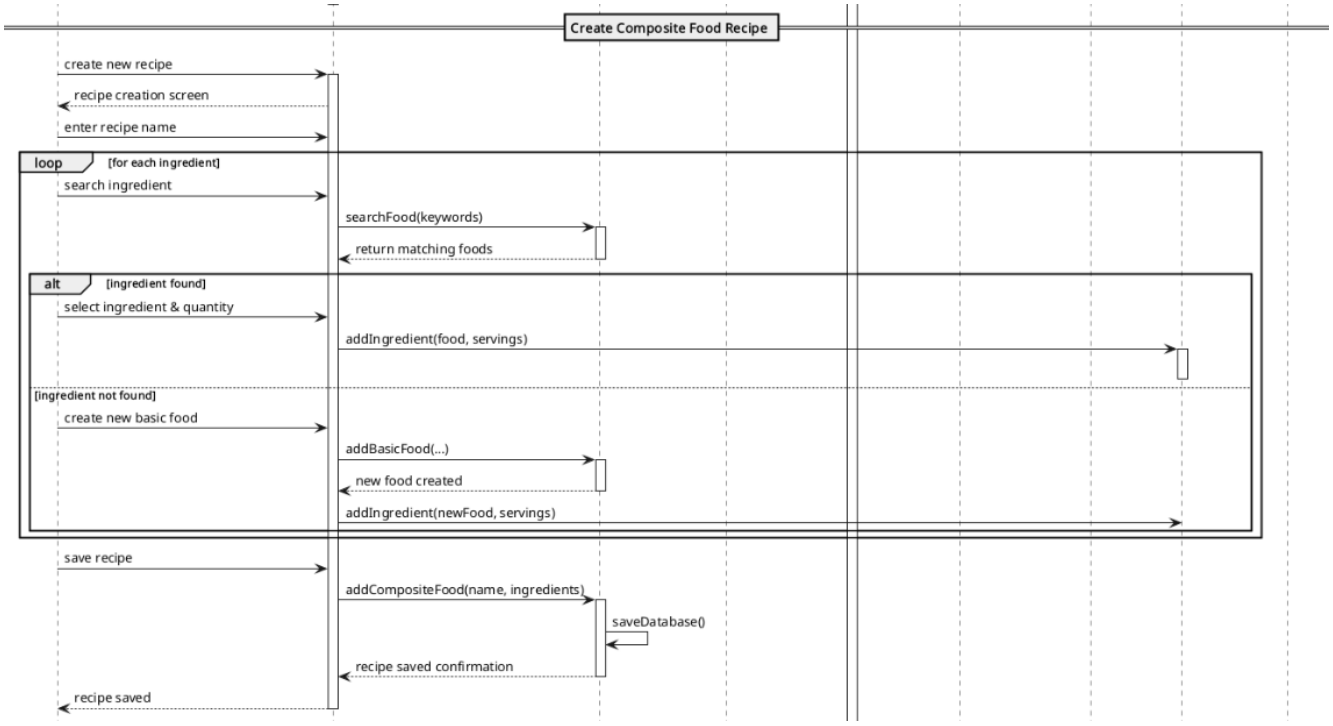
0. System Initialization



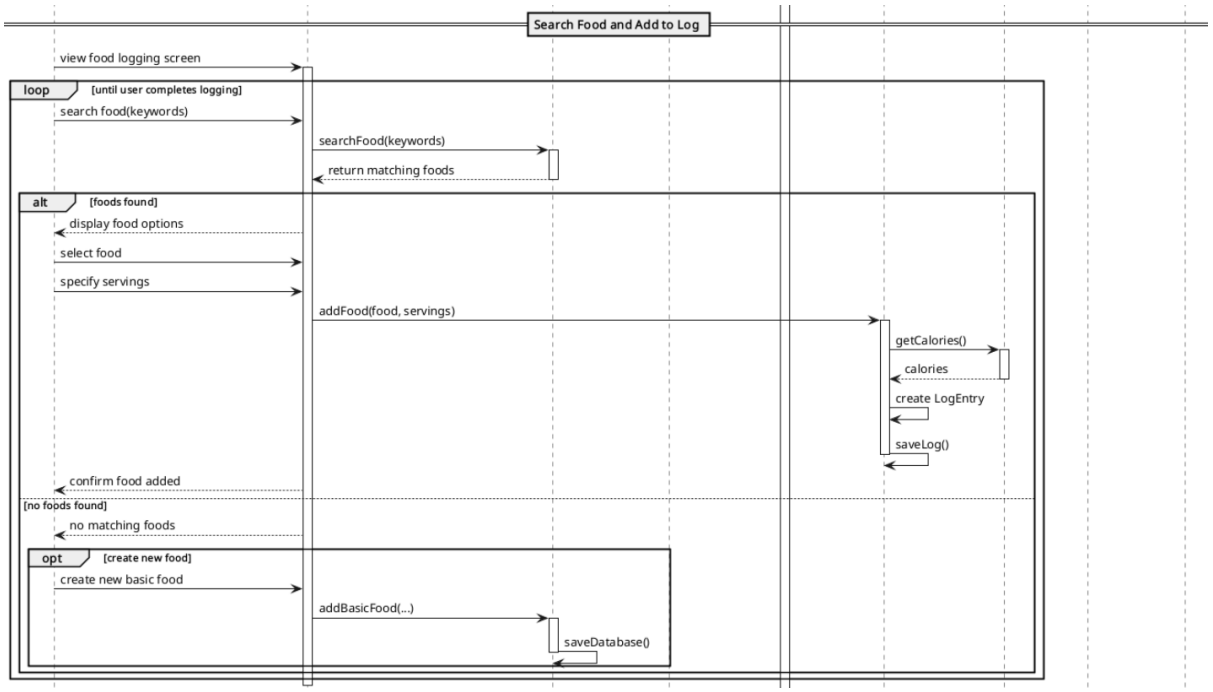
1. Adding a Basic Food to the Database



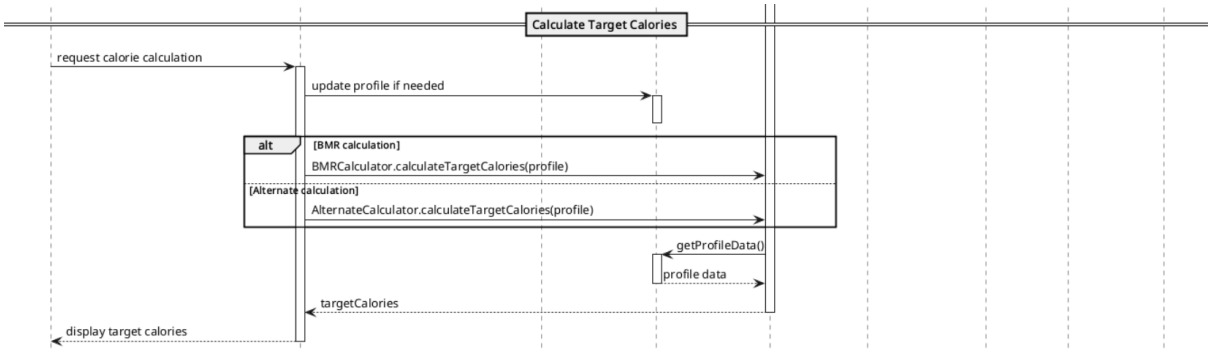
2. Creating a Composite Food Recipe



3. Logging Food Consumption

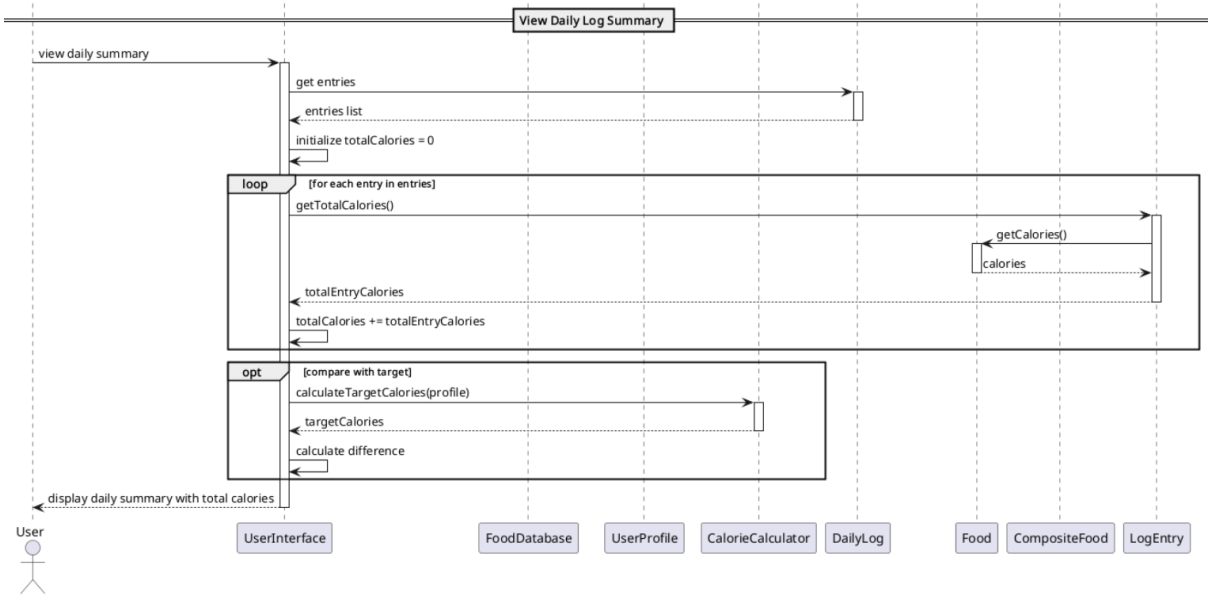


4. Calculating Target Calories



figureSequence Diagram: Calculating Target Calories

5. Viewing Daily Nutrition Summary



Design Narrative

The design of YADA reflects a balance among key principles of software engineering:

- **Low Coupling:** The use of interfaces (e.g., `CalorieCalculator`) ensures that components like calorie calculation methods can be swapped without affecting other parts of the system.
- **High Cohesion:** Each class has a single, well-defined responsibility. For example, `FoodService` handles food-related operations, while `LogService` manages daily logs.
- **Separation of Concerns:** The system separates concerns such as user profile management, food database operations, and calorie calculations into distinct modules.
- **Information Hiding:** Internal details of classes are encapsulated, exposing only the necessary methods to other components.
- **Law of Demeter:** Design minimizes direct interactions between unrelated classes, promoting modularity and maintainability.

Reflection

Strongest Aspects

1. **Extensibility:** The design supports adding new calorie calculation methods or food data sources with minimal changes.
2. **Undo Functionality:** The use of a stack in `DailyLog` ensures robust undo operations for log modifications.

Weakest Aspects

1. **Scalability of Food Database:** The current implementation may face performance issues with a large number of foods due to in-memory storage.

2. **Error Handling:** The error handling mechanisms for file operations (e.g., loading/saving logs) could be more robust.