Use Case Number: UC-01
Use Case Name: Add Basic Food to Database
Overview: This use case describes how a user adds a new basic food item to the system’s food database.
Actors: User (Primary), System
Precondition: The system is running, and the user has access to the system.

Flow:

Main (Success) Flow:

1. User selects the option to add a new basic food item.
2. System prompts the user to enter food details.
3. User provides:
    - Food name (identifier)
    - Keywords for search
    - Caloric value per serving
4. System validates the input.
5. System adds the new food item to the database.
6. System confirms the successful addition to the user.

Alternate Flows:
4a. Invalid Input: If the user provides incomplete or invalid details, the system prompts for corrections before proceeding.

Post Condition: The new food item is saved in the food database, and the user can search and use the food item in logs.

Use Case Number: UC-02
Use Case Name: Add Composite Food to Database
Overview: This use case describes how a user adds a new composite food item using existing food items.
Actors: User (Primary), System
Precondition: The system is running, and the food database contains at least one food item.

Flow:

Main (Success) Flow:

1. User selects the option to add a new composite food item.
2. System displays a list of available basic and composite food items.
3. User selects multiple food items and specifies their respective servings.
4. System calculates the total caloric value based on selected foods.
5. System prompts the user to enter an identifier and search keywords.
6. User provides the necessary details.
7. System saves the new composite food to the database.
8. System confirms the successful addition to the user.

Alternate Flows:
3a. No Available Food Items: If no food items exist in the database, the system informs the user and provides an option to add a basic food first.

Post Condition: The composite food is saved in the database, and users can search and log the composite food.

Use Case Number: UC-03
Use Case Name: Save Database Without Terminating Program
Overview: This use case describes how a user can manually save the food database without terminating the program.
Actors: User (Primary), System
Precondition: The system is running.

Flow:

Main (Success) Flow:

1. User selects the "Save Database" option.
2. System writes the current state of the food database to a file.
3. System confirms successful saving to the user.

Post Condition: The database is saved and can be loaded again later.

Use Case Number: UC-04
Use Case Name: Save Database Upon Program Termination
Overview: This use case describes how the database is saved automatically when the program terminates.
Actors: System
Precondition: The system is running.

Flow:

Main (Success) Flow:

1. System detects that the program is terminating.
2. System saves the current state of the food database to a file.
3. System confirms successful saving.

Post Condition: The database is updated before shutdown.

Use Case Number: UC-05
Use Case Name: Read Log on Program Start
Overview: This use case describes how the system loads the food consumption log when the program starts.
Actors: System
Precondition: The system is starting, and a previous log exists.

Flow:

Main (Success) Flow:

1. System reads the existing log file.
2. System loads the food entries into memory.

3. System confirms successful loading.

Post Condition: The food log is available for use.

Use Case Number: UC-06
Use Case Name: View, Select, and Update Log
Overview: This use case details how a user can view, select, and update logged food entries.
Actors: User (Primary), System
Precondition: The system is running, and the user has logged food entries.

Flow:

Main (Success) Flow:

1. User selects the "View Log" option.
2. System displays logged food entries for a selected date.
3. User selects an entry to update.
4. User modifies the food item or servings.
5. System updates the log.
6. System confirms the update.

Post Condition: The log entry is updated successfully.

Use Case Number: UC-07
Use Case Name: Record and Update User Profile Information
Overview: This use case describes how the system records and updates user profile details.
Actors: User (Primary), System
Precondition: The system is running, and the user has an existing profile.

Flow:

Main (Success) Flow:

1. User selects "Update Profile."
2. System displays:
    - Gender
    - Age
    - Height
    - Weight
    - Activity level
3. User updates any required information.
4. System saves the changes.

Post Condition: The user profile is updated and stored.

Use Case Number: UC-08
Use Case Name: Compute Target Calorie Intake

Overview: This use case describes how the system calculates the user's target calorie intake.
Actors: System
Precondition: The user has an active profile.

Flow:

Main (Success) Flow:

1. System retrieves the user's profile data.
2. System calculates the target calorie intake using a selected method.
3. System displays the calculated intake.

Post Condition: The user can see their recommended calorie intake.

Use Case Number: UC-9
Use Case Name: Compare Calories Consumed vs. Target
Overview: This use case describes how the system compares actual food intake with the user’s calorie target.
Actors: User (Primary), System
Precondition: The user has logged food entries, and the system has a target calorie intake calculated.

Flow:

Main (Success) Flow:

1. System retrieves total calories consumed for the selected date.
2. System compares total intake with the target calorie intake.
3. System displays:
    - Total calories consumed
    - Target calorie intake
    - Difference (positive or negative)

Post Condition: The user is informed about their calorie balance for the day.