SWEN 383 - Software Design Principles and Practices

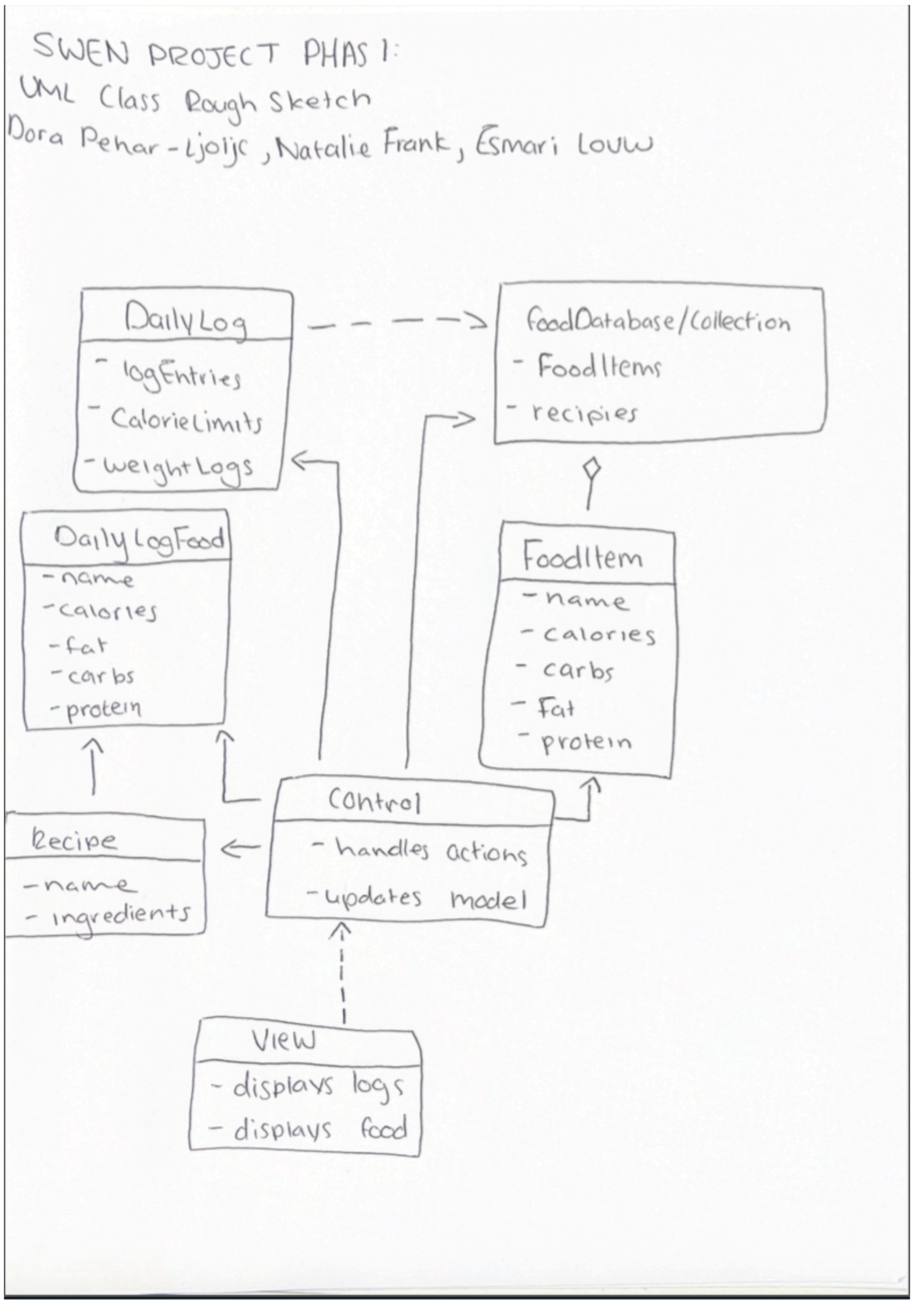
DESIGN SKETCH

Version 1.0

Date: 01/04/2025
Team Identification: Group 4
Team Members:

- Esmari Louw
- Dora Pehar-Ljoljic
- Natalie Frank

Rough Design Class Sketch:



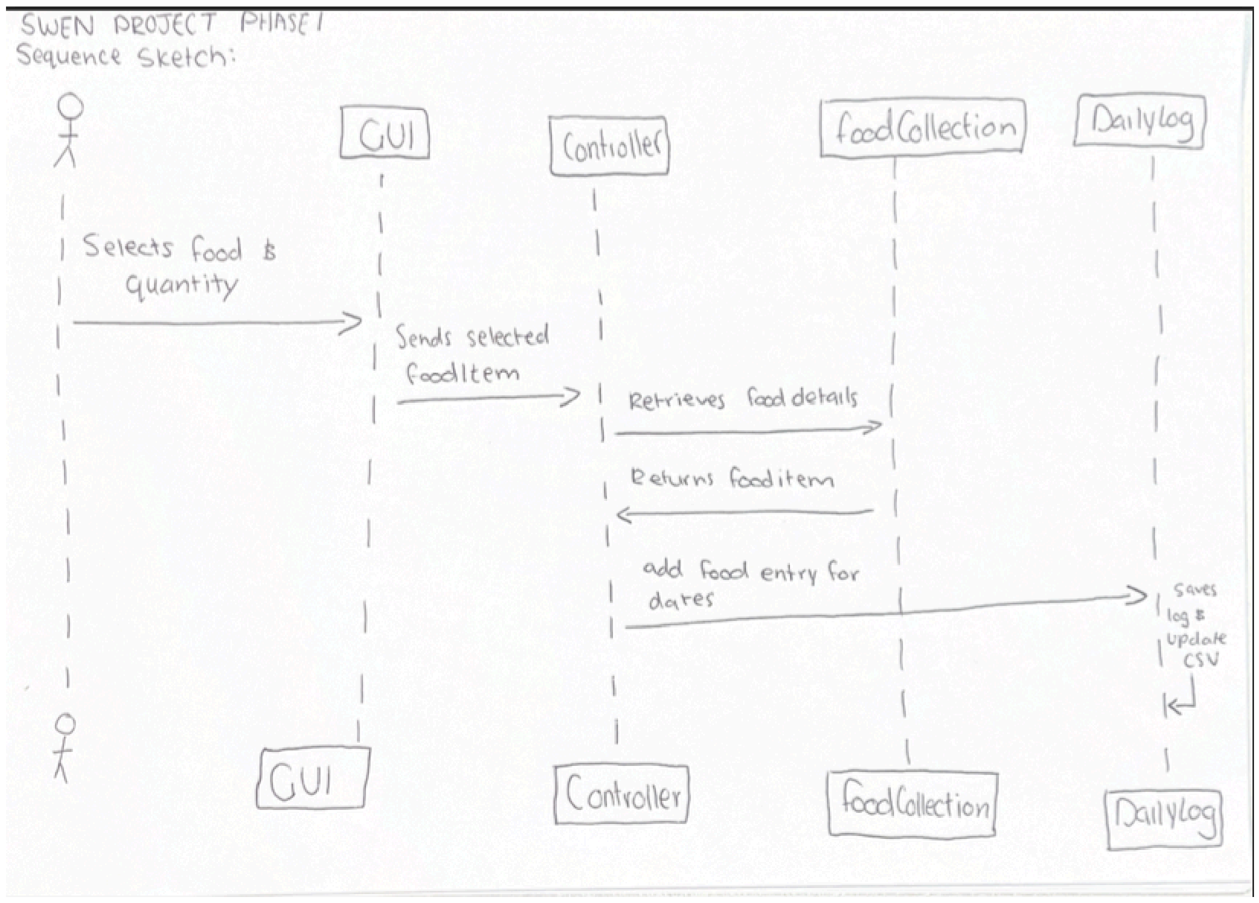
Key Classes:

- **DailyLog :** Handles food logging for each day, manages the log entries, calorie limits and the weight records
- **DailyLogFood :** Represents the individual food item with nutrition details
- **Recipe:** Special type of food composed of multiple ingredients/components
- **FoodCollection:** Manages the storage and retrieval of available food items.
- **Controller:** handles the applications logic and interaction between UI and data
- **Views:** Represents the graphical user interface to display logs and user inputs.

Design Patterns Used:

- **Adapter pattern** : to integrate FoodItem and Recipe under a unified DailyLogFood interface
- **Composite Pattern:** used in FoodCollection to allow treating individual FoodItems and Recipes uniformly
- **MVC Pattern:** separates concerns between Model (DailyLog & FoodCollection), View (GUI) and Controller (Control class).
- **Factory pattern:** used in GUI for creating UI components dynamically.

Rough Sequence Diagram:



Reading the Food Data:

- The system reads a food collection FoodItem and Recipe
- Recipes can contain other food items or recipes.
- FoodCollection loads and Stores the data
    - Objectives involved = Control -> FoodCollection -> DailyLogFood / Recipe

Adding Food to Daily Log:

- The user selects a food item from collection
- The selected item is added to the DailyLog for the current date.
- The system updates and saves the log.
    - Objects involved = View -> Control -> DailyLog -> DailyLogFood

Computing total Calories:

- User requests total calories computation for given date
- The system retrieves logged food entries and calculates the total calories
- If calorie limit is set, the system compares it to the logged intake

- Objects involved = View -> Control -> DailyLog -> DailyLogFood

Class Descriptions:

- **DailyLog:** Stores food entries per date, tracks calorie limits and weight.
- **DailyLogFood:** Represents a single food item with calories, fat, protein and carbs.
- **Recipe**: Inherits from DailyLogFood, containing multiple DailyLogFood entries
- **FoodCollection:** Manages food items and recipes, allowing access to stored data.
- **Controller:** Manages logic and connects UI with the data model.
- **View:** Displays information and interacts with the user.

Our Design Rationale:

Advantages:

- Modular, keeping flexibility for future (phase II) modifications.
- The Adapter Pattern unifies the recipes and food items.
- The MVC pattern makes it easy to modify UI without affecting core logic in the program.
- The Composite Pattern allows uniform treatment of individual foods and recipes.
- The factory Pattern allows for scalable GUI component creation.

Disadvantages:

- Initial complexity of handling the recipes as nested structures.
- Large project = more complex structures of files.
- MVC introduced more components to manage, requiring proper coordination.
- Requires efficient handling of the I/O operations.

Next Steps:

- Refine Class Diagram (will be uploaded into folder on GIT-Repository)
- Improve Sequence diagram (will be uploaded into folder on GIT-Repository)
- Skeleton Code = uploaded through pushes in our git history
- GUI integration