Criterion B: Design

Contents

Design Overview

overview of diet generator program.[1]

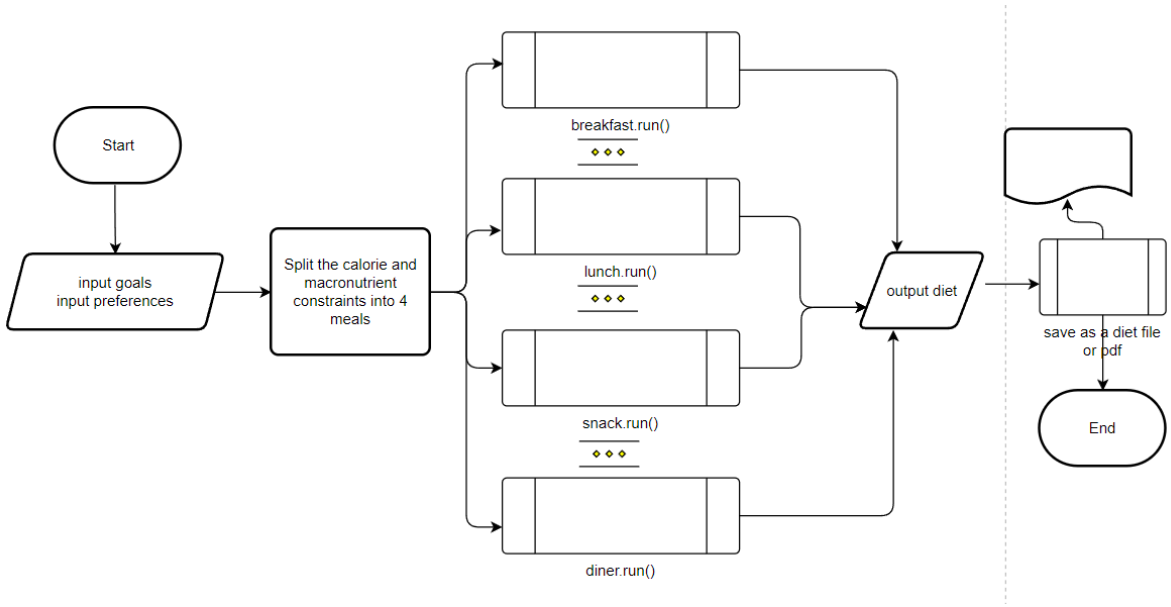

Figure 1. Overall view of diet generator[2]

Prototype GUI Designs

Figure 2. Staring window[3]

[1] The genetic algorithms are inspired, in part, by research paper *Diet Generator Using Genetic Algorithms* by researchers Catal´an-Salgado Edgar-Armando, Zagal-Flores Roberto, Torres-Fernandez Yuliana, and Paz-Nieves Alexis

[2] Software for creating flow charts: https://online.visual-paradigm.com/

[3] Software for creating GUI Prototypes: https://figma.com/ and https://excalidraw.com/

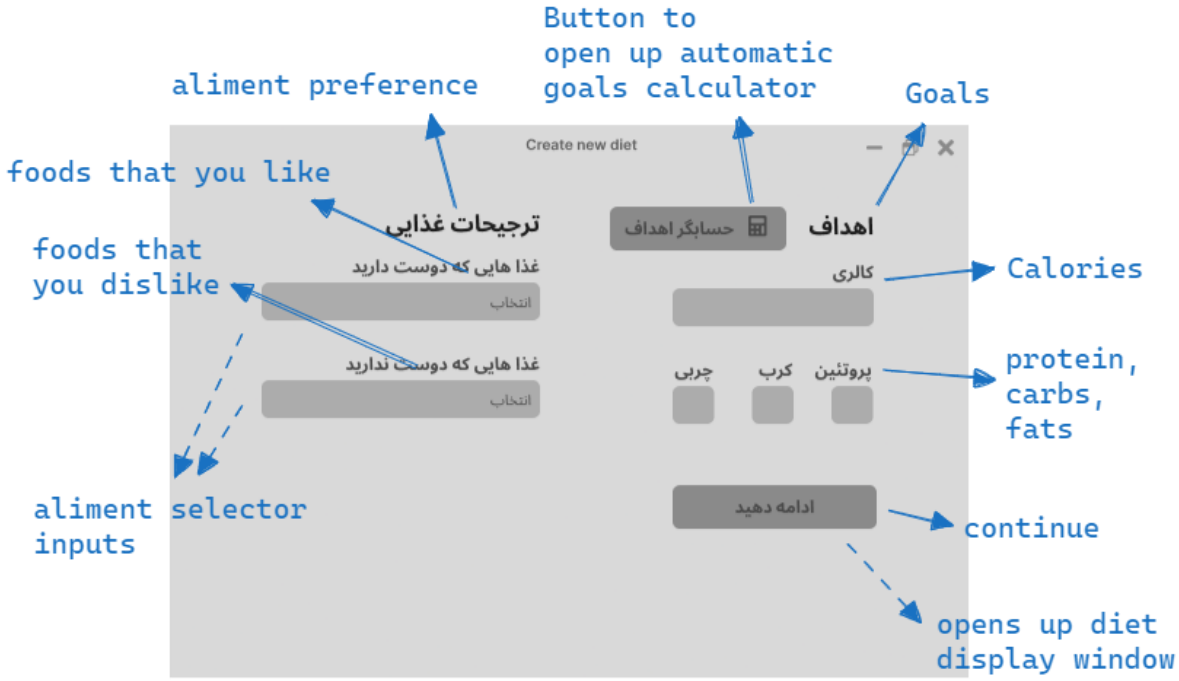

Figure 3. diet creation window

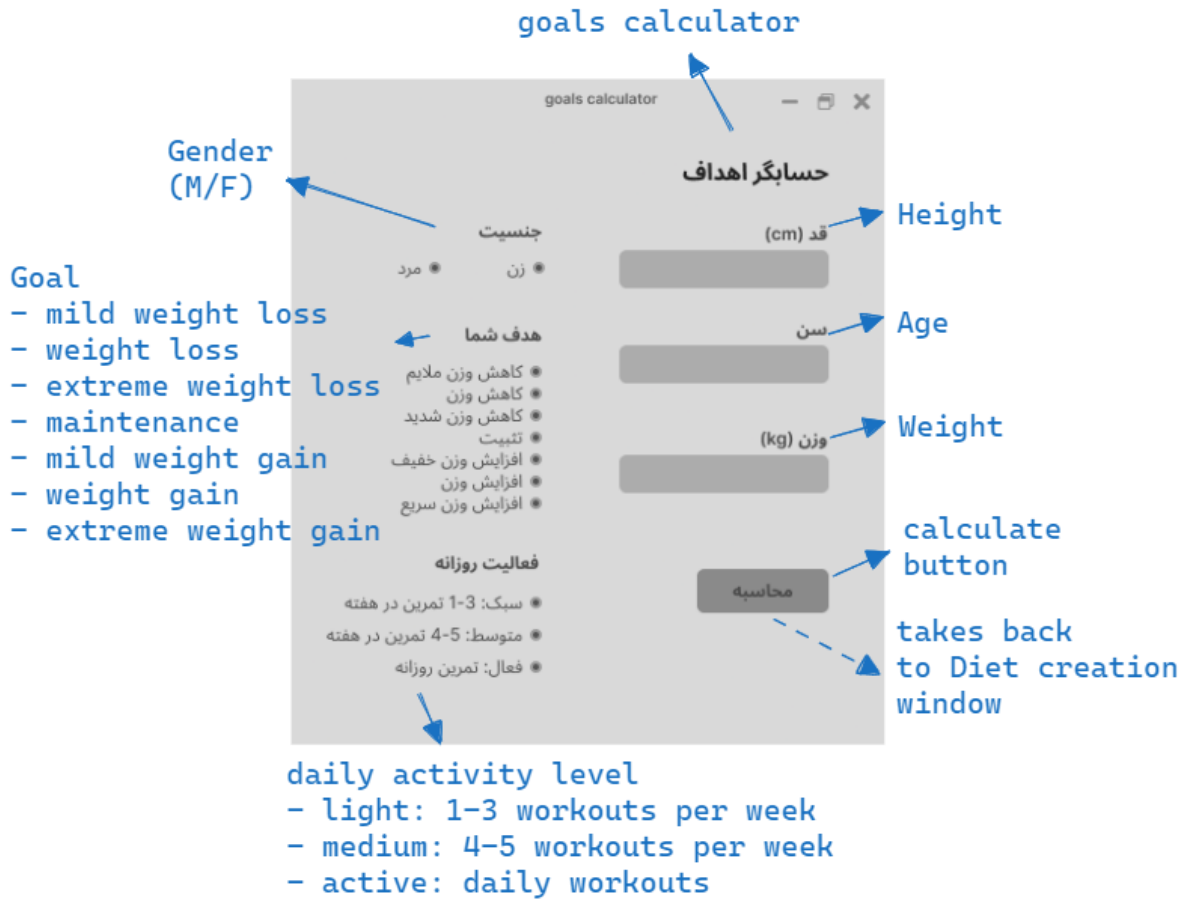

Figure 4. goals calculator window

The goals calculator window automatically fills out the calorie and macronutrient inputs through simpler data such as Height, age, etc.

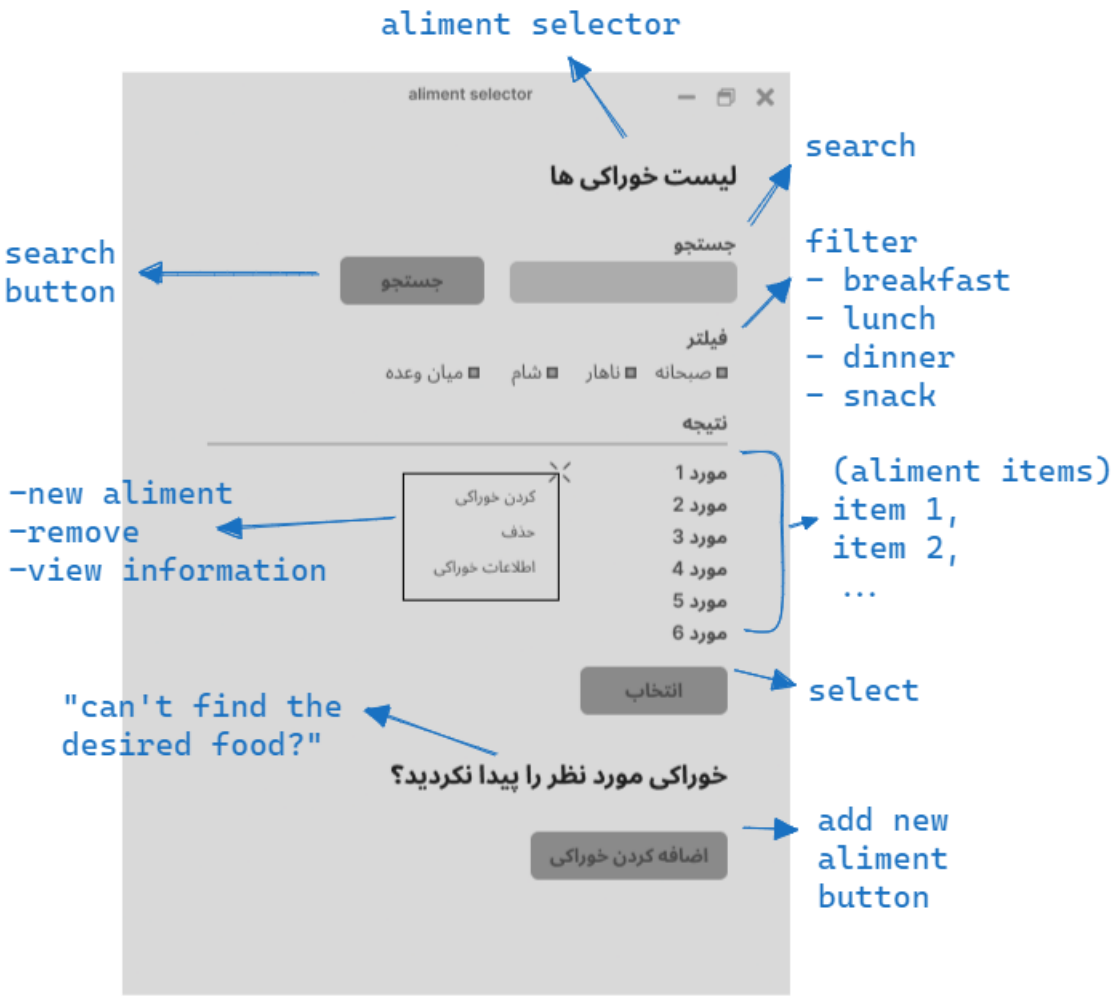

Figure 5. aliment selector window

opens whenever user is required to select a food item. If food item does not exist in file, the user can add it.

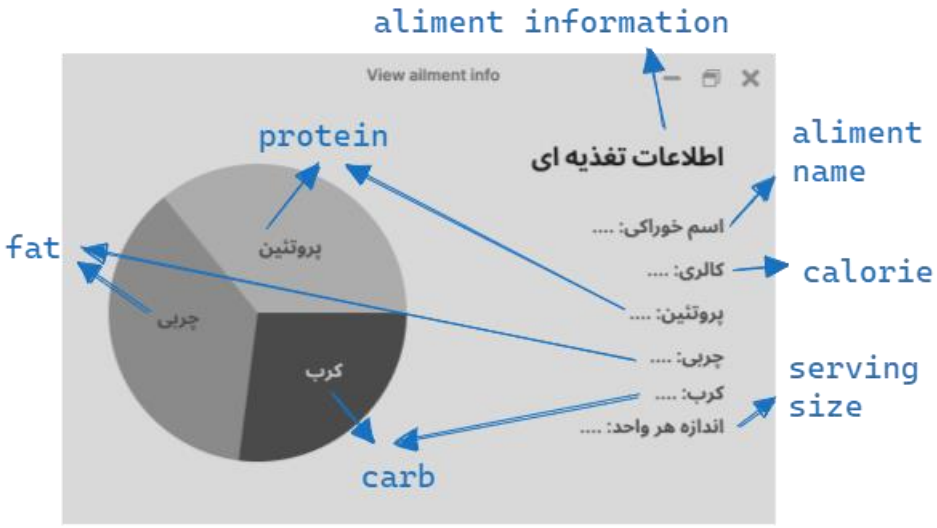

Figure 6. view aliment info window

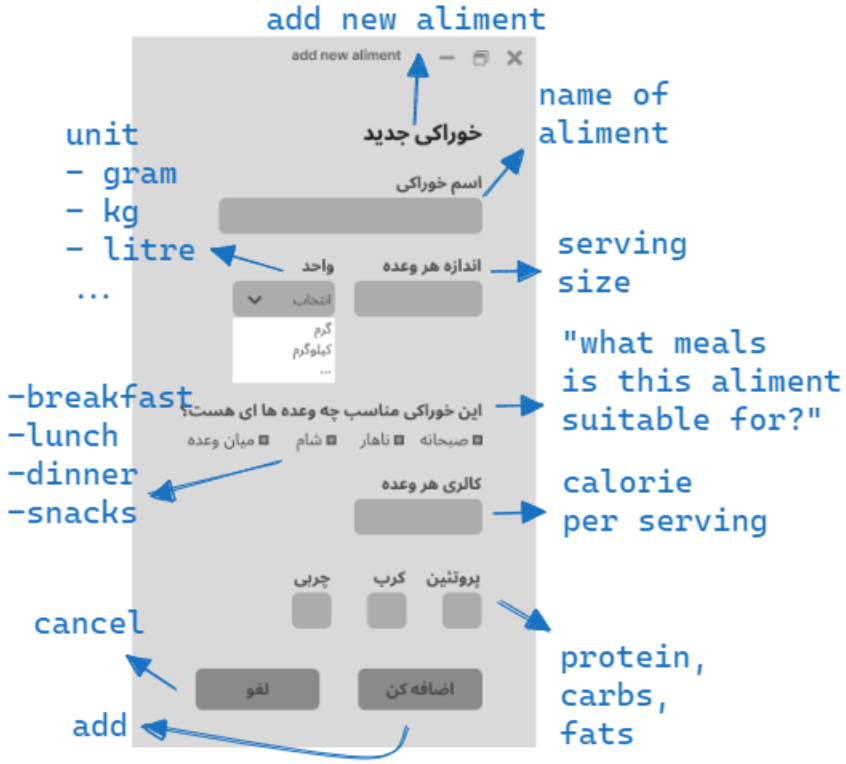

Figure 7. add new aliment window

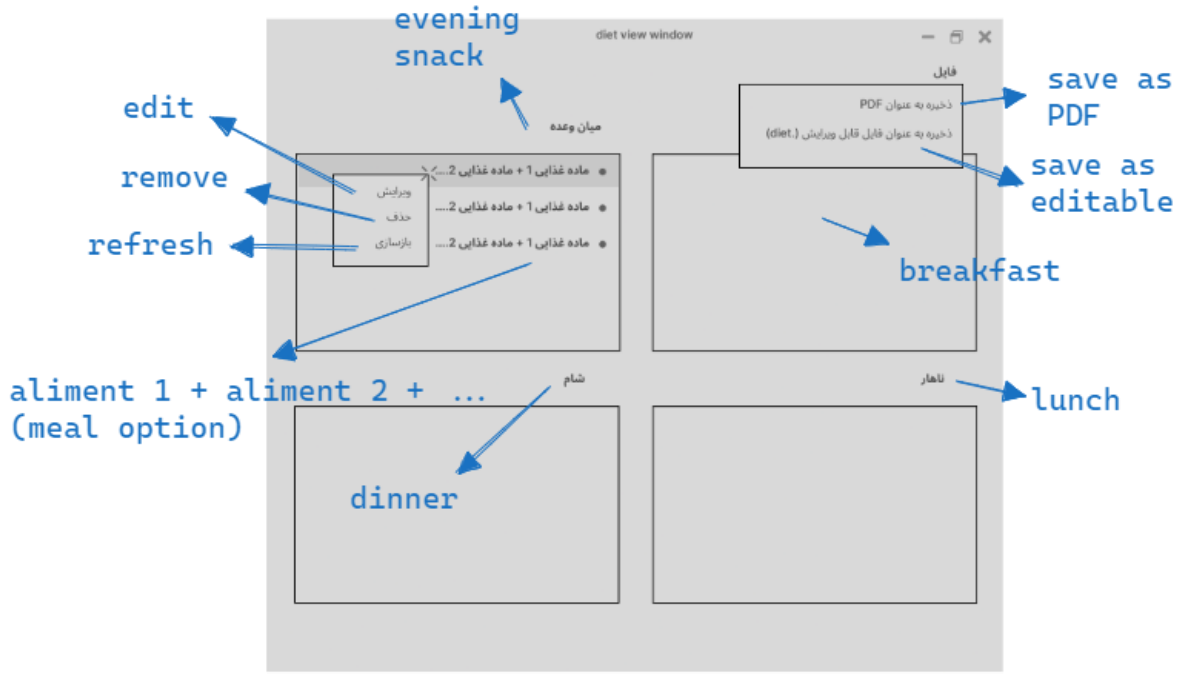

Figure 8. diet display window

Once the diet has been created, it will be displayed in this chart. Each bullet point is a meal option.

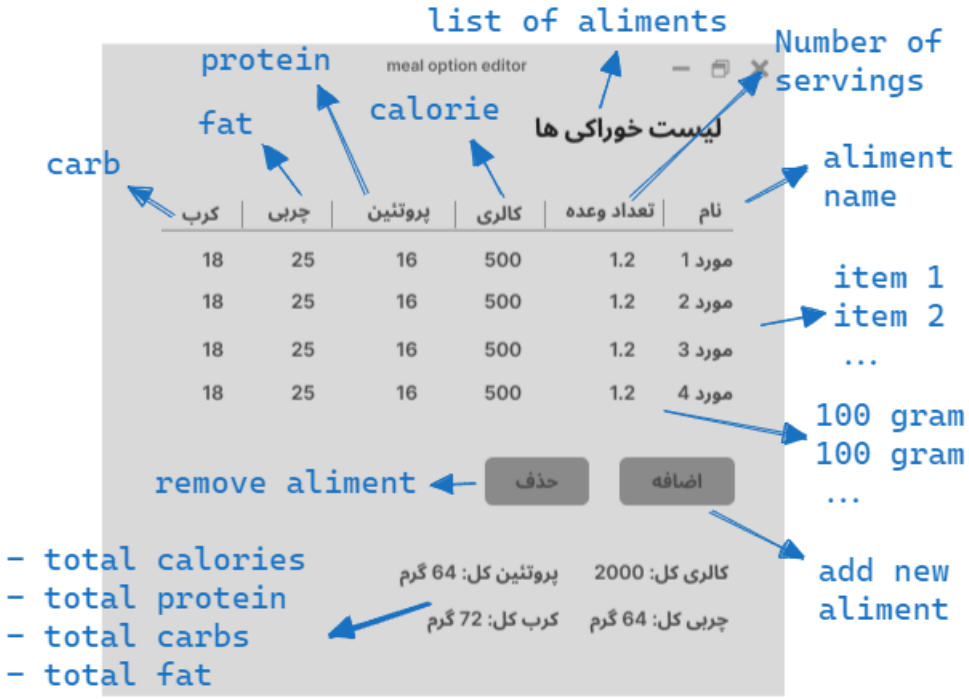

Figure 9. meal option editor window

is used to edit a meal option. If the client is not satisfied with a meal option, he can manually add or remove aliments. The table displays the aliments and their amount in each meal option.

Data File Structure

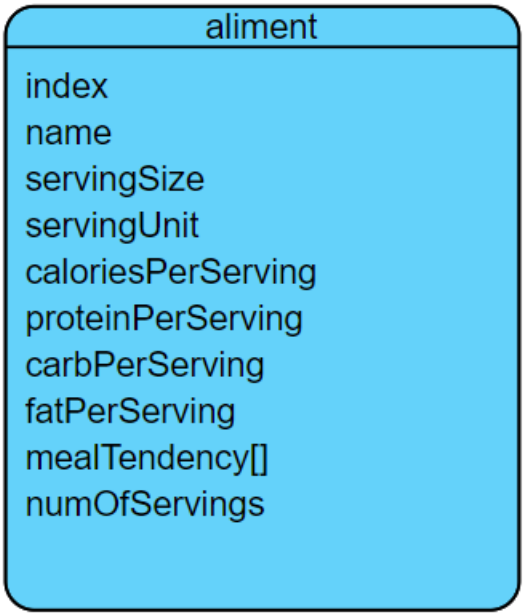

Figure 10. Data file structure[4]

mealTendency is a collection which specifies which meals the aliment is suitable for. mealTendency is important to consider, to prevent algorithm from recommending mismatched meal options. E.g., it would be unreasonable to suggest chicken as a breakfast option, or milk as a lunch option.

[4] Software for creating data file structure: https://online.visual-paradigm.com/

UML Diagrams

GUI UML

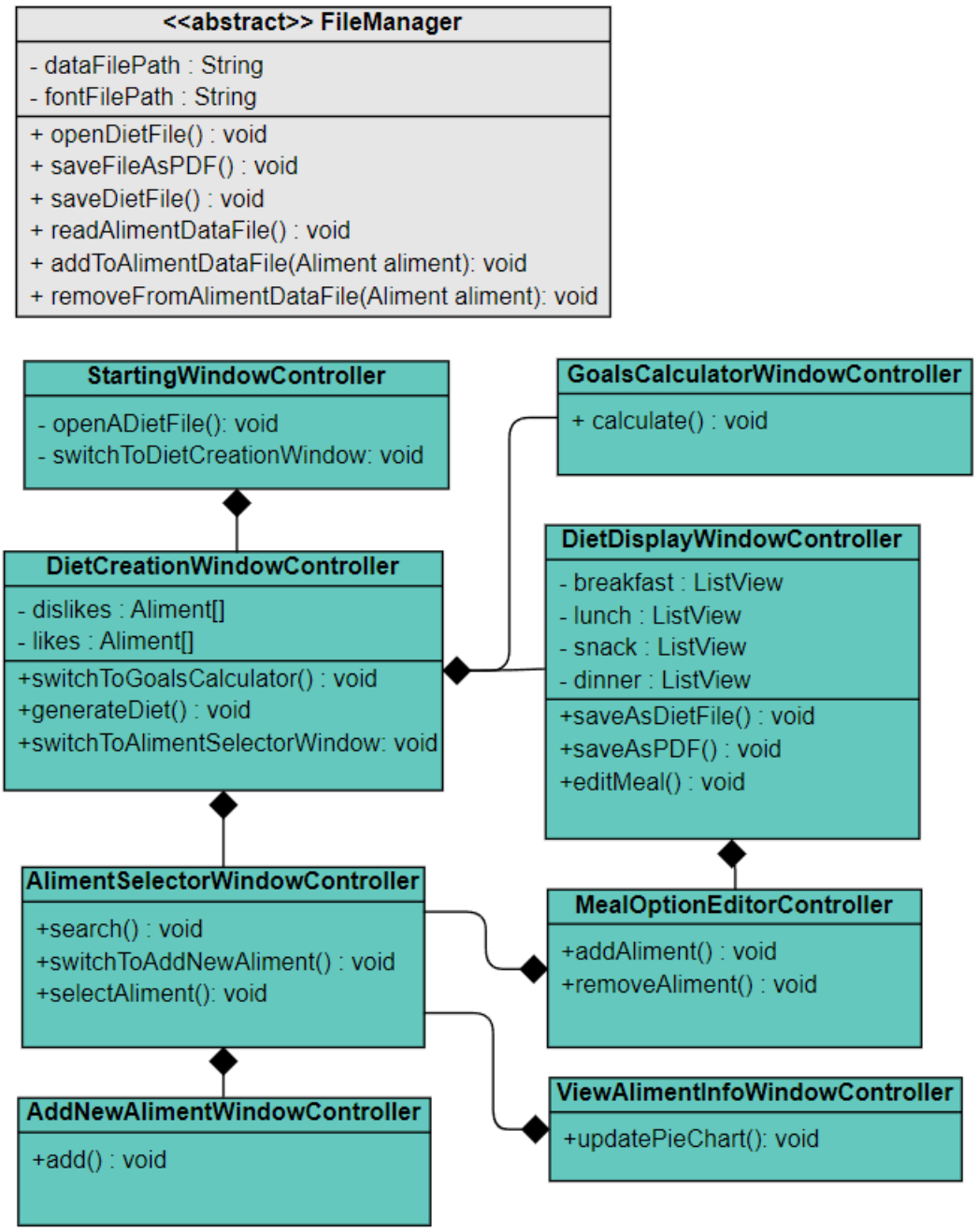

Figure 11. GUI UML[5]

[5] Software for creating UML: https://online.visual-paradigm.com/

Each class represents a controller for the GUI windows discussed in *Prototype GUI Designs* section. The GUI elements have been ommited from this UML to prevent it from becoming crowded.

Genetic Algorithm UML

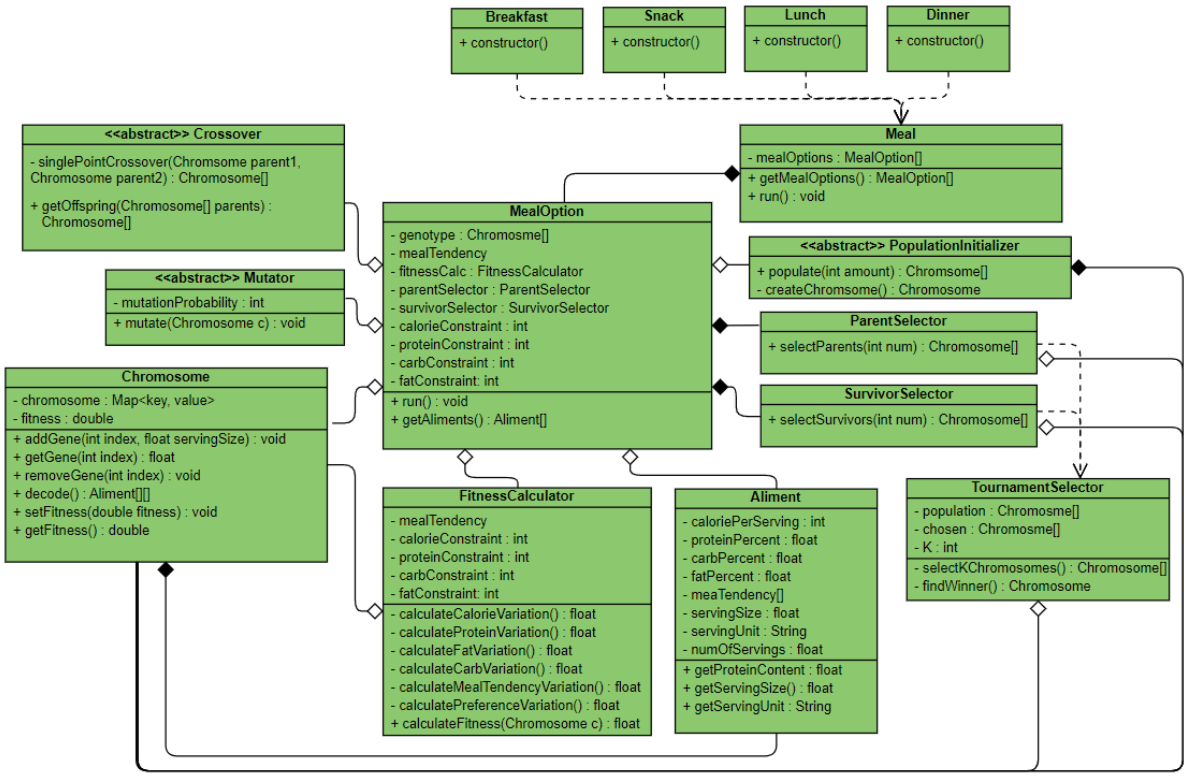

Figure 12. GA UML

Meal: superclass of four meals breakfast, lunch, snack, dinner. Contains 5 MealOptions

MealOption: is where the genetic algorithm is run to create a suitable meal option

Aliment: contains nutritional information about food item.

Chromosome represents a solution, or meal option, as an encoded genotype. The genotype is in computational space. Once the decode method is called, genotype is decoded into phenotype. Phenotype is an array of Aliment[], and represents a possible meal option.

PopulationInitializer: creates a semi random initial population for genetic algorithm

FitnessCalculator contains methods necessary to calculate the fitness of a chromosome.

ParentSelector and **SurvivorSelector** inherit from the **TournamentSelector** class since the algorithms are similar.

Crossover: creates offsprings based on parent pairs

Mutator: mutates a chromosome with a predetermined probability.

Flow Charts

Genetic algorithm

A genetic algorithm is run for each meal option. Each meal has 5 meal options, and there are 4 meals. Hence the genetic algorithm is run 20 times on different threads.

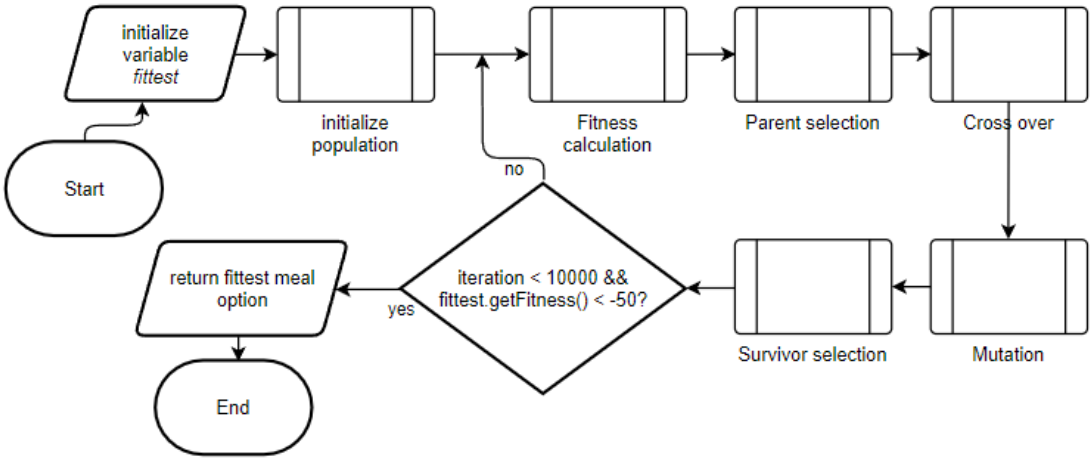

Figure 13. Genetic algorithm in meal option class overview

Population Initialization

With a predetermined chance (E.g., 20%), select food from preferred list. This takes user preference into consideration. Otherwise, the initial population is random.

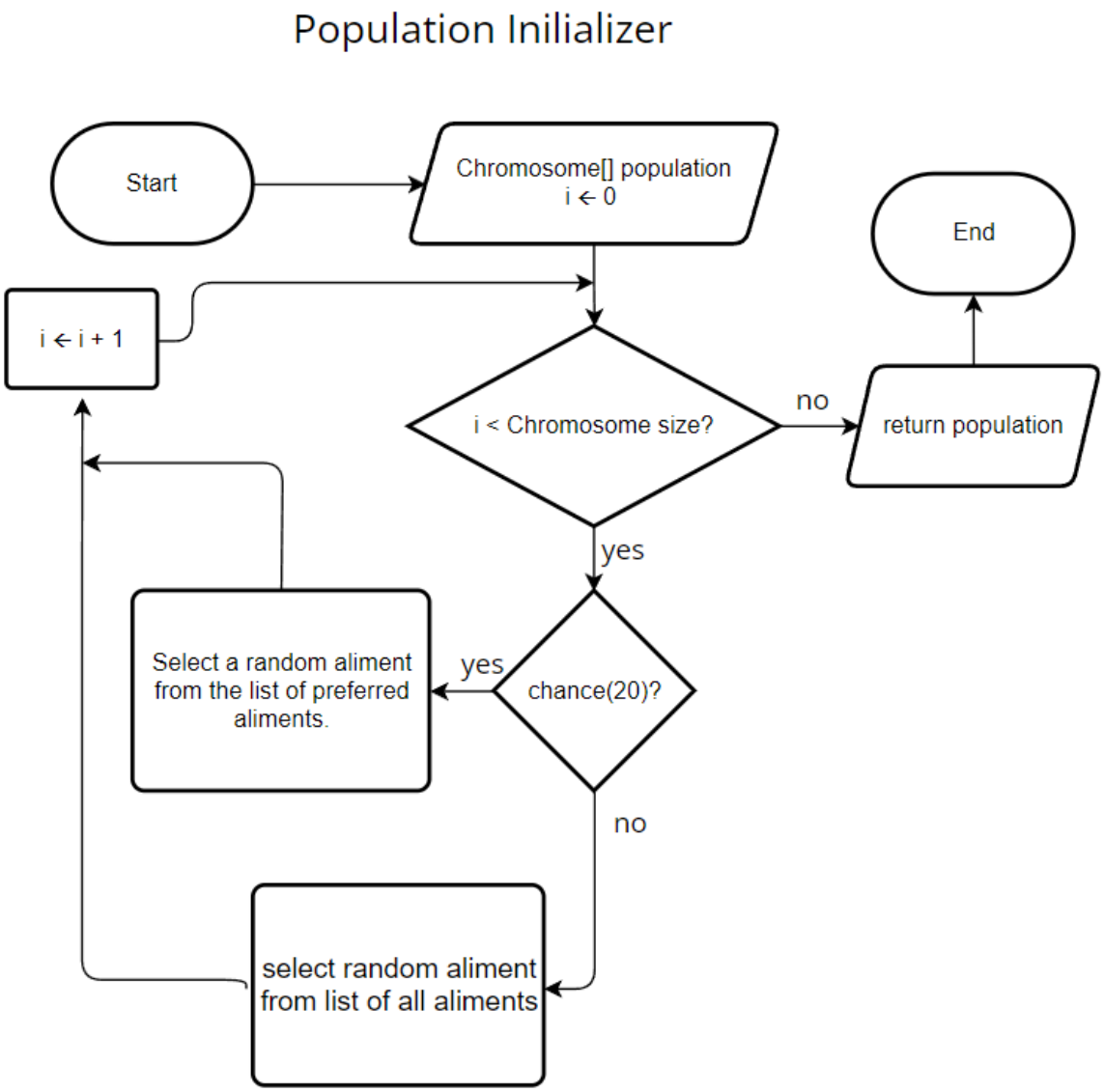

Figure 14. Population initialization

Fitness calculation

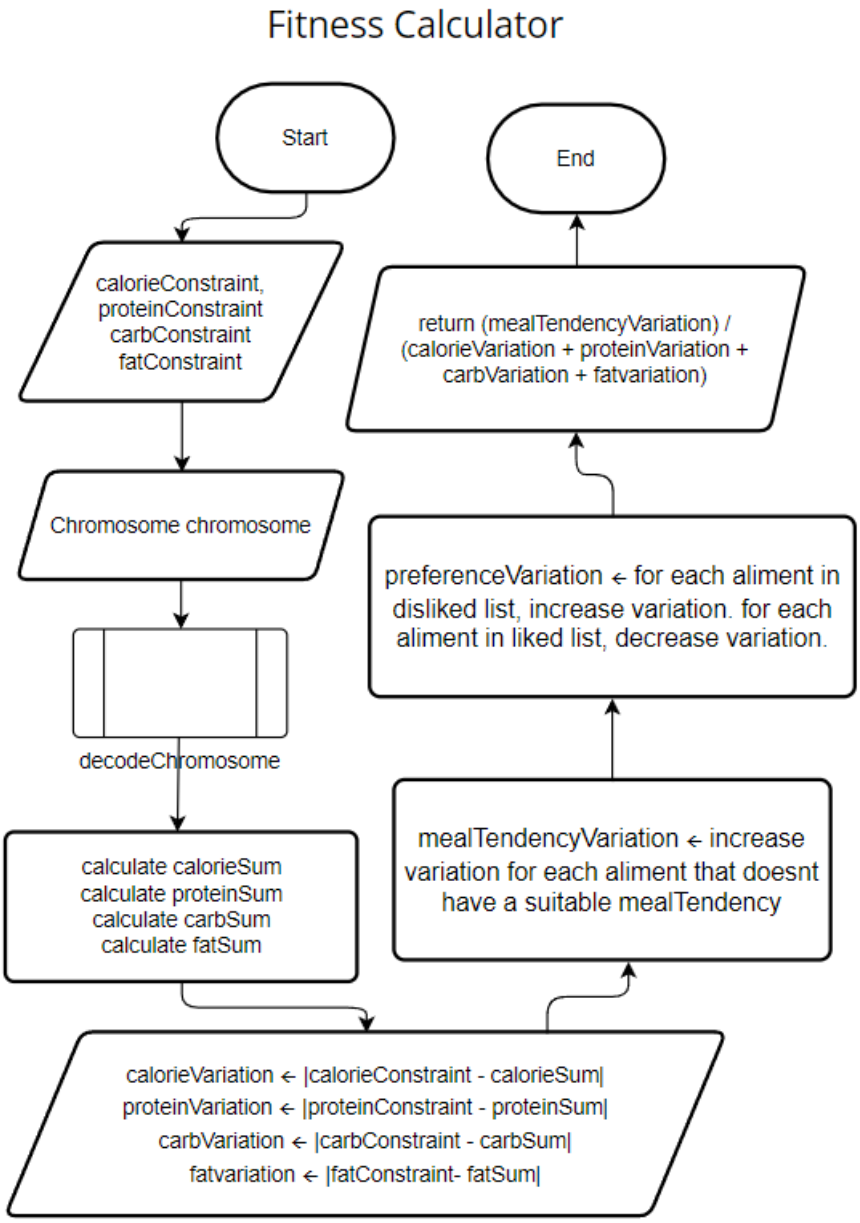

Figure 15. Fitness calculation

Parent selection and survivor selection

Parent selection and survivor selection are mentioned together because they both implement K-means tournament selection algorithm

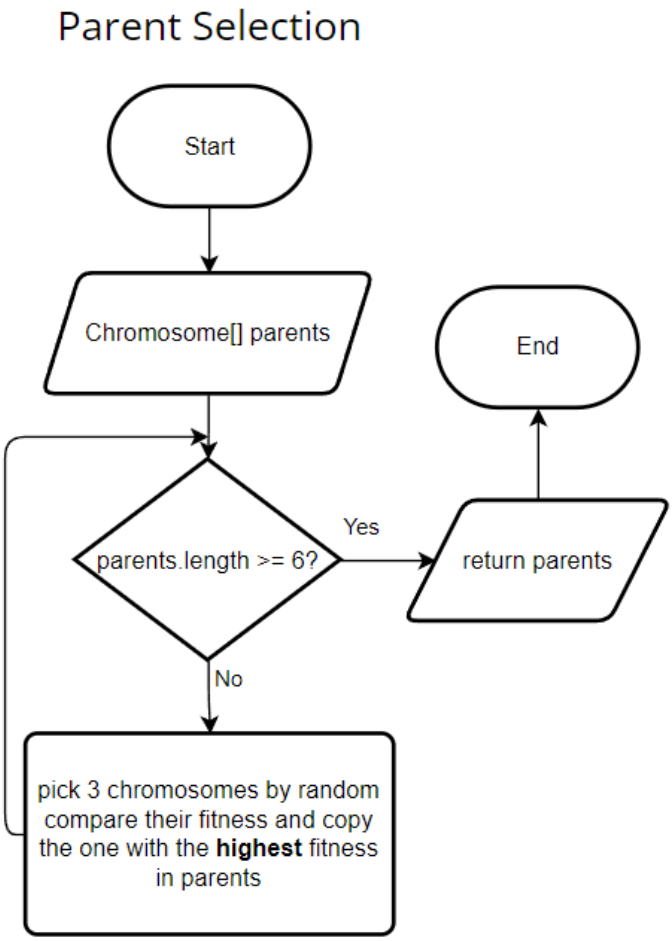

Figure 16. implementation of tournament parent selection to choose 6 parents with K equal to 3

Cross over

- Two parents with high fitness are chosen for offspring creation
- Single point cross over algorithm: random index is generated, and genes of each parent are swapped.

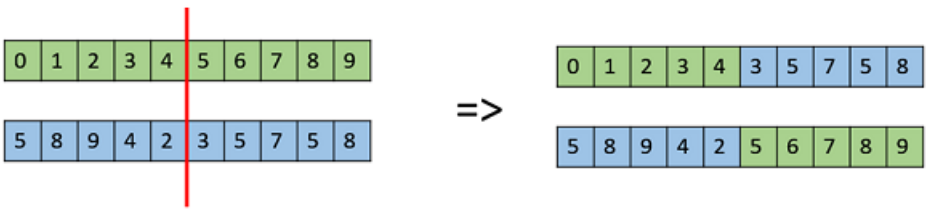

Figure 17. Single point crossover[6]

[6] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

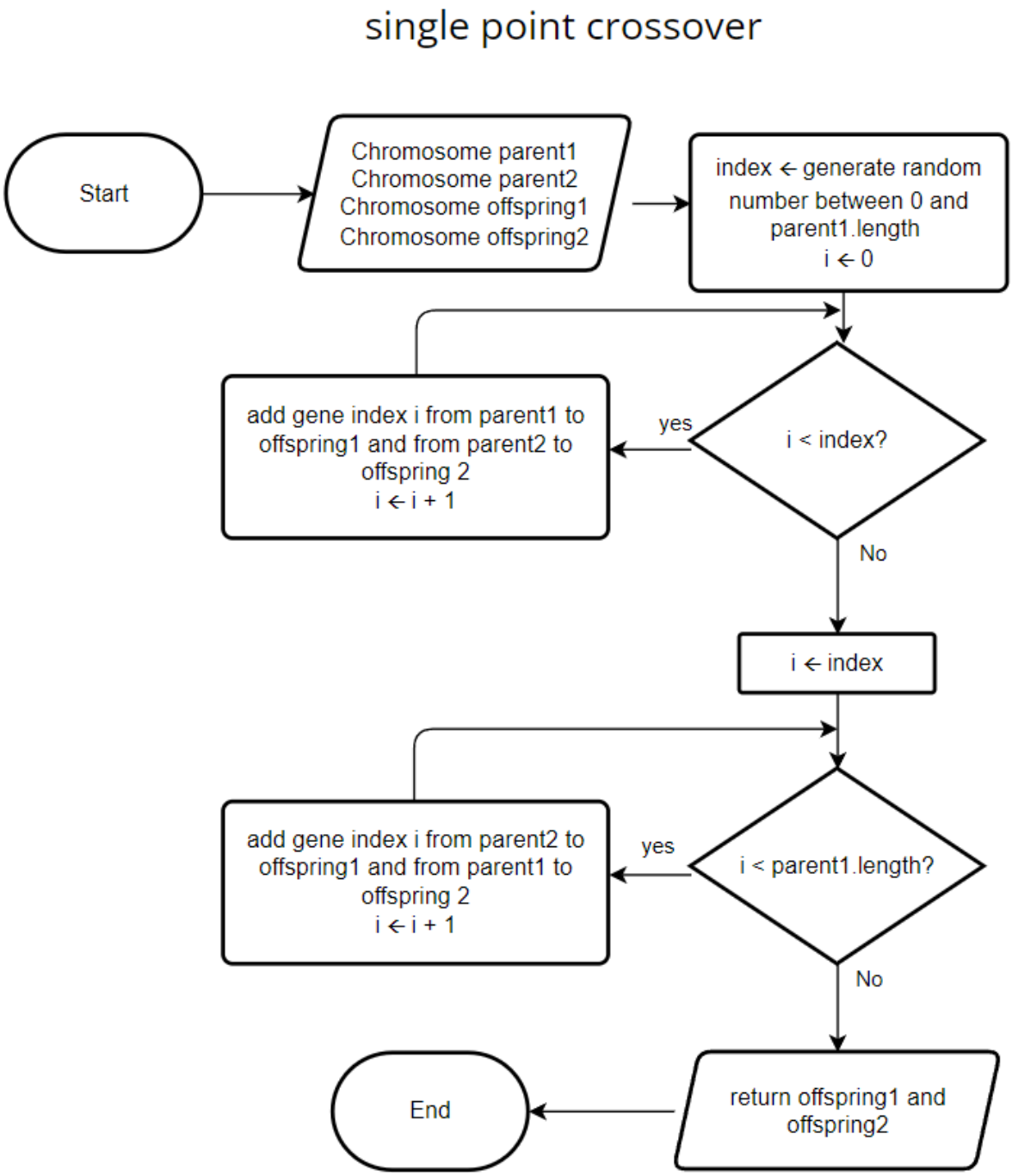

Figure 18. single point cross over algorithm flowchart

Mutation

- Creates variety
- Prevents elitism
- One gene of chromosome is randomly changed.
- Happens with low probability

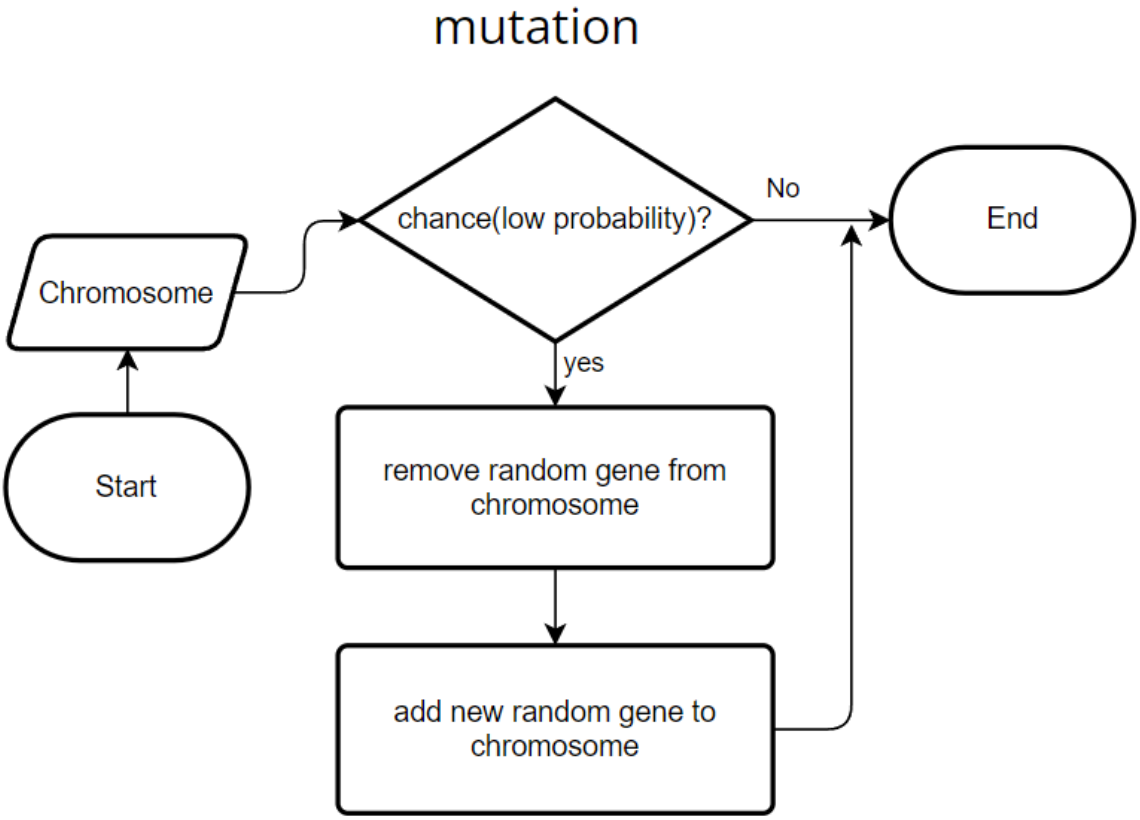

Figure 19. mutation

Test plan

Action to be tested	Method of testing
Check whether *alimentSelectorWindow* properly selects aliments, and the search and filter functionalities work as intended.	Open the window and filter for a certain meal (ex. Breakfast) and check if results are filtered. Then search for a food, next click the selection button and check whether it is selected
Check if macronutrient data is correctly validated	Enter different macronutrients and see if appropriate alerts are displayed.
Check if *goalCalculatorWindow* calculates the goals correctly	Input different data and check accuracy of output with online goal calculators.
Check if program can read from datafile	Use *aliment selector* window to check whether data is read from file.
Check if program can write to the datafile	Use *addNewAlimentWindow* to check whether data is added to file
Check if program can remove aliments from datafile	Use the remove button on *alimentSelectionWindow* to see if aliments are correctly removed.
Check whether the calorie of the generated diets is in line with the calorie constraints.	Use different food preferences and goals and check whether the calorie of the generated diet is consistent with the calorie constraint.
Check whether meal options can be manually modified.	Use *MealOptionEditor* to check whether user can add/remove aliments from a meal option.
Check whether the macronutrients of the generated diets are in line with the macronutrient constraints.	Use different food preferences and goals and check whether the macronutrients of the generated diet are consistent with the macronutrient constraint.
Check if details of nutritional data are correctly displayed	Use *ViewAlimentInfoWindow* to see whether aliment information is displayed, and whether the data is correct

Check diets can be saved as editable files.	Generate a diet and save as an editable file. Next, check whether the saved file can be loaded and edited in the program.
Check whether diet contains the food preference of the user	Input random foods in the food preference inputs, and then generate a diet and compare the similarity
Check if diets are correctly saved as PDFs	Generate a random diet, and press save as PDF. Check whether the file has been created and open it to see if the contents are same as the generated diet.