Software Engineering

Books or notes are **not** allowed.
Write only on these sheets. **Concise** and **readable** answers please.

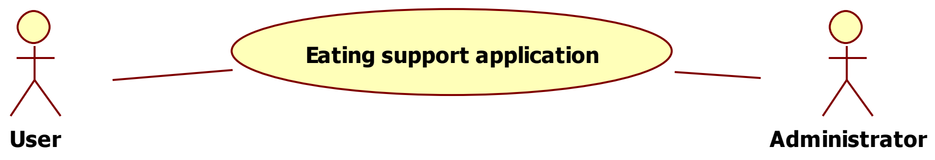
Surname, name, matricola ____

Eating support application
Many applications are already available to support people in monitoring their eating.
Each time the user eats something she can use the application to log what she has eaten. There are at least three ways of doing this. The user looks for the food eaten (ex: hamburger and fries, or pasta carbonara etc) in a predefined list; then the user can customize the quantity of the food eaten (ex 150g pasta carbonara). Or the user enters the quantities of basic ingredients eaten (butter 10grams, sugar 15 grams, etc). The user can customize the lists, adding foods and ingredients.
Or the user selects a meal. A meal is made of a number of foods (ex hamburger and fries + muffin, or pasta carbonara + green salad). The application provides a list of meals.
The user can also define and add complete meals to the list of meals.
Having logged what he has eaten over time, the user can monitor his eating habits, in various ways: calories eaten per week, per day, per meal. Number of meals per day and time interval between them. Basic nutrients (carbohydrates, saturated fat, non saturated fat, proteins, vitamins, minerals, etc) per week, per day, per meal.
Besides the above monitoring function, the application allows the user to compare her eating habits with some predefined habits. A habit is defined in term of RDA (Recommended Dietary Allowance) per calories and per basic nutrient (ex RDA protein = 150 g, RDA saturated fat 20g, etc). Again the application provides some predefined good eating habits customized in function of sex, age, height and living style (athlete, manual worker, non manual worker etc); plus the capability of defining new habits.
The application could also allow the user to define a monthly /weekly/ daily diet. A diet is made of foods. Then the user can compare her eating habits with the defined diet.

The application requires the user to define an account, and should be usable both by a smartphone and a PC.

In the following you should analyze and model the eating support application.
1 (15 points) – a. Define the context diagram (including relevant interfaces)

Actor	Physical interface	Logical interface
User	Screen keyboard (touchscreen)	GUI
Administrator	Screen keyboard (touchscreen)	GUI



The app most likely has a client and a server part (this is a design choice to be made later, but since large parts of the data are common to many users client server is a better choice). In any case at this level the server is inside the context diagram. The app could also require an email server to send / receive emails (in this case email gateway is an actor).

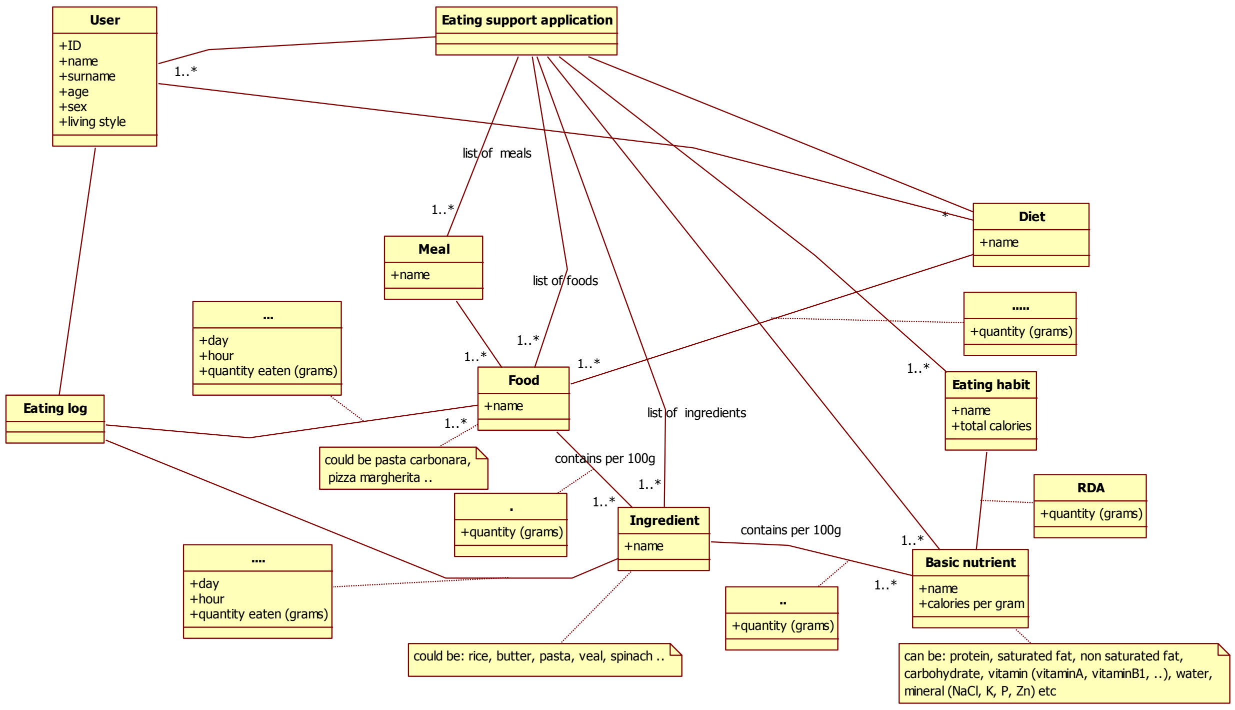
Define the key concepts and entities and their relationships (UML class diagram) for the application

The classes Meal, Food, Ingredient, Basic Nutrient contain what is possible to eat, at different detail levels, the lowest level being the Basic nutrient. The relationship between one level and the next details the quantities involved, (ex Ingredient == butter, 100g butter contain 90g saturated fat and 10g water). This structure is static and does not change (apart foods or meals possibly added by the user): 100g butter always contain the same quantity of fat and water., 100g pasta carbonara always contain x gram pasta and y gram eggs. Given a meal, or a food, or an ingredient, and a quantity ,it is possible to compute the basic nutrients associated with the quantity, following the associations.
Class Eating log allows to model what a user eats, in terms of Food and Ingredient. Quantity eaten, and when, are written in the association class between Eating log and Food / Ingredient. Ex someone eats 50g butter on july 7 10 am. A class could be used instead of an association class, but the key point is that the quantity (and date) depend both on the user and the Food /Ingredient, so it is wrong to place the quantity on Food (or on User).

Class Eating habit defines the RDAs in terms of basic nutrients (ex an Eating habit.name == athlete could require 3000 calories, with 300g carbohydrates, 100g fat, etc..).
Class Diet defines a number of Foods to be eaten , and the quantity per each. Diet could define similar associations with Meal and Ingredient (not inserted to keep the diagram cleaner).

Remark that classes Eating Log, Eating habit, Diet, all repeat the same pattern with classes Meal Food, Ingredient.

All comparison requested (ex Eating log vs Habit, or Eating log vs Diet) can be obtained by following the associations (in other words they are computed when needed).


User
+ID
+name
+surname
+age
+sex
+living style
Eating support application
Diet
+name
Meal
+name
Food
+name
Ingredient
+name
Basic nutrient
+name
+calories per gram
Eating habit
+name
+total calories
RDA
+quantity (grams)
Eating log
...
+day
+hour
+quantity eaten (grams)
....
+day
+hour
+quantity eaten (grams)
.....
+quantity (grams)
.
+quantity (grams)
..
+quantity (grams)
list of meals
list of foods
list of ingredients
contains per 100g
contains per 100g
1..*
*
could be pasta carbonara, pizza margherita ..
could be: rice, butter, pasta, veal, spinach ..
can be: protein, saturated fat, non saturated fat, carbohydrate, vitamin (vitaminA, vitaminB1, ..), water, mineral (NaCl, K, P, Zn) etc

List the requirements in tabular form (do not forget to list important NF requirements)

ID	Type (Functional Non Functional)	Description
1	F	**User** management: CRUD (Create, Read, Update, Delete) User, authorize and authenticate User
2	F	**Eating items** management: CRUD Meal, attach/detach Food to meal, CRUD Food, attach / detach Food to Ingredient, CRUD Basic nutrient, attach/detach Ingredient to Basic Nutrient
3	F	**Eating log** management. CRUD Eating log, attach/detach Eating log to User, attach detach Eating log to Meal / Food / Ingredient
4	F	**Diet** management: CRUD Diet, attach / detach Food / Meal to Diet
5	F	**Eating Habit** management: CRUD Eating Habit, attach / detach to basic Nutrient
6	F	**Analyze Eating Log**: compute total calories / basic nutrients per meal / day/week/month
7	F	**Analyze Diet**: compute total calories / basic nutrients
8	F	**Compare Eating Log – Diet:** call 6, call 7, compare calories, basic nutrients
9	F	**Compare Eating log – Eating Habit:** call 6, compare calories, basic nutrients
10	NF	Privacy: all data of a User should be visible to the user only

Define one scenario describing a user who logs one meal
Precondition: user U is already registered, meal M is not logged to U

Postcondition: meal M is logged to U

Step	Description	Req ID
1	User U logs in	1
2	User U is authorized	1
3	U retrieves meal M from list of meals	2
4	U attaches M to his Eating log, setting day and quantity	3
5	U logs out	1

2 (7 points) -Define black box tests for the following class, using equivalence classes and boundary conditions.

double computeCaloriesFood(int weightProteins, int weightFats, int weightCarbohydrates)

The function receives the weight in grams of proteins, fats, carbohydrates in a food and computes the total amount of calories given. 1g protein = 4calories, 1g carbohydrates = 4 calories, 1g fat = 9calories.

Ex. computeCaloriesFood (1,2,3) → 1x4 + 2x9 + 3x4 = 34calories

WeighProteins	weightFats	weightCarbohydrates	Valid / Invalid	Test cases
[minint, 0[	[minint, 0[	[minint, 0[	I	T(-10, -10, -10, err)
		[0, maxint]	I	T(-10, -10, 100, err)
	[0, maxint]	[minint, 0[	I	T(-10, 10, -10, err)
		[0, maxint]	I	T(-10, 10, 10, err)
[0, maxint]	[minint, 0[	[minint, 0[	I	T(10, -10, -10, err)
		[0, maxint]	I	T(10, -10, 10, err)
	[0, maxint]	[minint, 0[	I	T(10, 10, -10, err)
		[0, maxint]	V	T(1, 2, 3, 34)

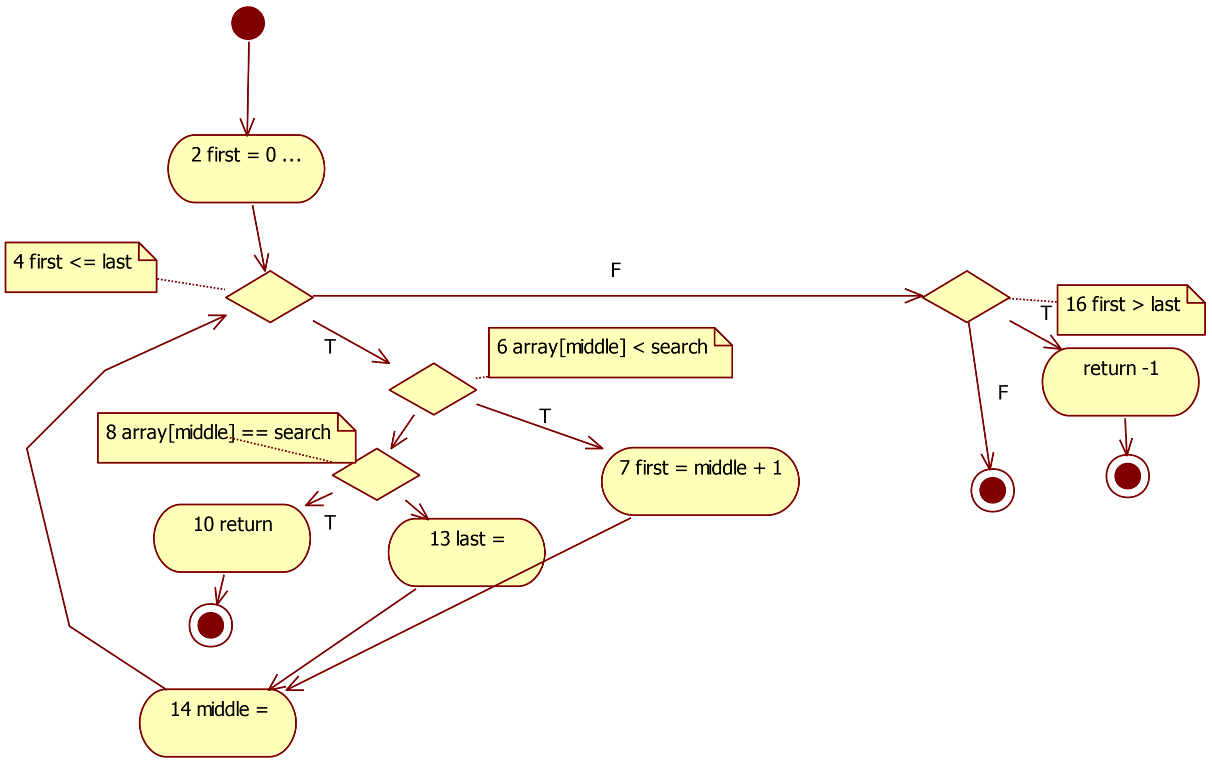
Boundary test cases (not exhaustive)
T(0,0,0, 0)
T(-1,0,0, err)
T(1,0,0, 4)

3 (7 points) – For the following function define the control flow graph, and define test cases to obtain the highest possible node coverage, edge coverage, multiple condition coverage, loop coverage, path coverage.
For the test cases, **write only the input value**.

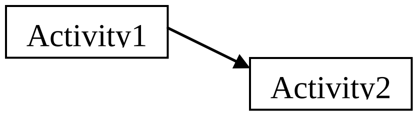
```
int binarySearchArray(int* array, int n, int search) {
  first = 0;    last = n - 1;    middle = (first+last)/2;

  while( first <= last )
  {
     if ( array[middle] < search )
        first = middle + 1;
     else if ( array[middle] == search )
     {
         return (middle+1);
     }
      else
     { last = middle - 1; }
      middle = (first + last)/2;
   }
   if ( first > last )
      return -1;
 }
```

Coverage type	Number of test cases needed to obtain 100% coverage	Coverage obtained with test cases defined (%)	Test cases defined
Node	<=3	100%	T1({1,3,5,7,8,9,11},7, 8) found T2({1,3,5},3, 99) not found
Edge	<=4	100%	T1 T2
Multiple condition	No multiple conditions	-	-
Loop	3	100%	T3({1},0, 2) zero loop T4({1},1, 2) one loop T5({1,3,5},3, 2) many loops
Path	100% path coverage not feasible in practice		


2 first = 0 ...
4 first <= last
F
T
6 array[middle] < search
T
8 array[middle] == search
T
10 return
13 last =
7 first = middle + 1
14 middle =
16 first > last
T
F
return -1

4 (1 points) – A project has the following estimated Gantt.



Planned value of the project is 100 units, 50 on activity1, 50 on activity2. One month after start the project has consumed 60 units, activity1 is finished, activity2 is not. What is the Earned Value of the project?
EV = 50. Activity2 does not count because is NOT finished

5 (1 points) – "Cost of the project should be measured in Euros" what kind of requirement is this?

Domain requirement (a kind of NF requirement)

6 (1 points) – Describe shortly the Delphi estimation method.

See slides

7 (1 points) – What is the recommend length of an iteration in an agile project?

4 -5 weeks at most

8 (1 points) – Describe shortly the 'Strategy' Design pattern

See slides