FACULTY OF COMPUTING
UTM Johor Bahru

Semester II 2023/2024

Subject : Programming Technique 2 - SECJ1023

Section : 08

Task : Problem Analysis and Design

Due : 29 May 2024

Lecturer : DR LIZAWATI BINTI MI YUSUF

Group : 08

	Name	**Matric Number**
1	RAMI YASSEIN ELTAYEB MOHAMED	A23CS0022
2	Ammar Abdulrahman Anaam Mudhsh	A23CS0287
3	Mohamed Ali Mohamed Ali	A21EC0287

Table of Contents

1.0 Section A...

1.1 Flow Chart 2-3

1.1 about flowchart 4

2.0 Section B...

2.1 Objects...6

2.2 Classes involved 7-9

2.3 Classes relationship..10

3.0 Section C...

3.1 Class Diagram(UML 11

.

1.0 Section A

1.1 Flowchart :-

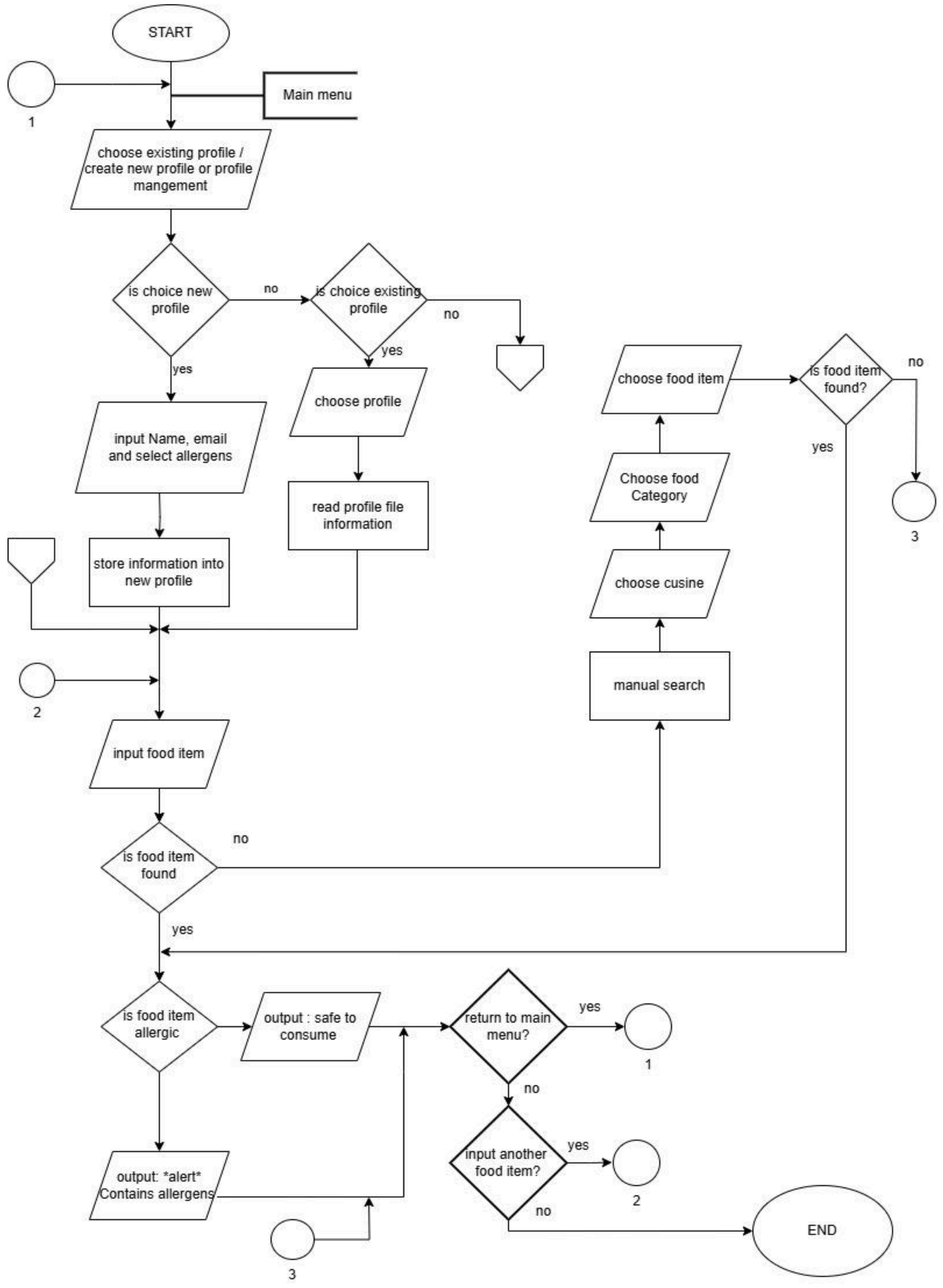


Figure 2

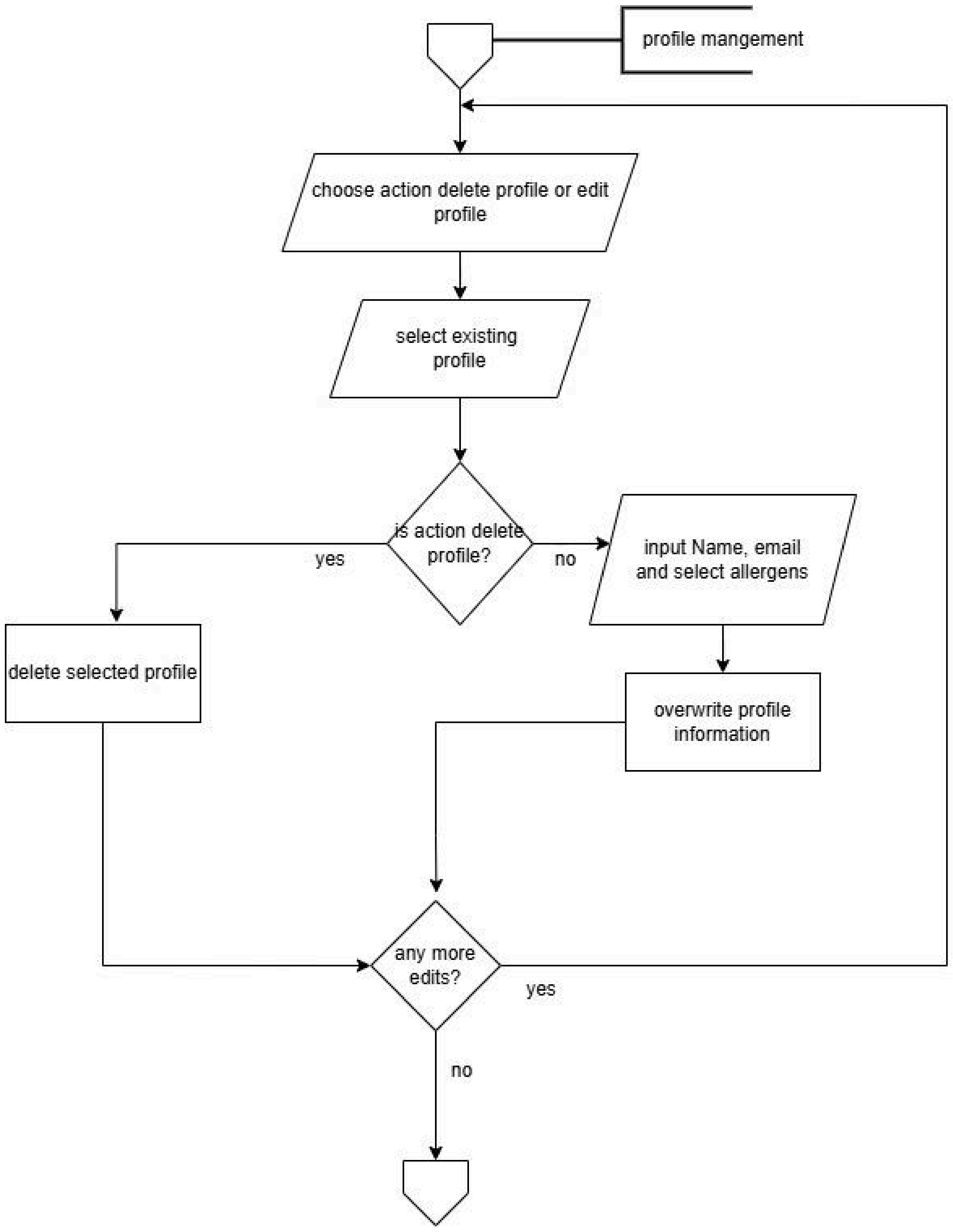

profile mangement
choose action delete profile or edit profile
select existing profile
is action delete profile?
yes
no
input Name, email and select allergens
delete selected profile
overwrite profile information
any more edits?
yes
no

Figure 2

<u>**1.2 About the flowchart:-**</u>

The main process of the system begins with the Main Menu, where users can choose to manage their profiles or search for food items. Users can create a new profile by providing personal details such as name, email, age range, and selecting allergens from a pre-populated list. If an existing profile is chosen, the system retrieves the stored information for the user to review or update.
Once a profile is set up, users can perform food searches. They can either use the Search Bar to type in specific food items or browse through categorized lists of foods. The system checks the ingredients of the selected food against the user's allergy profile to determine if it is safe to consume. If the food contains allergens, the system alerts the user, specifying the allergens found. If no allergens are detected, the system confirms the food is safe.

The system also provides detailed reporting based on search results, informing users whether a food is "Safe to Consume" or "Contains Allergens" along with specifics. Additionally, users can search by allergen to find foods that contain or are free from the allergen and those that may pose a cross-contamination risk. Throughout the process, users can update their profiles to ensure accurate and personalized allergy checks, continually enhancing their experience and safety.

2.0 Section B (Problem Analysis)

2.1 Objects:

1. **User:** Represents a user of the system with attributes like name and allergies.
2. **Allergen:** Represents a specific allergen (e.g., peanuts, dairy) with an attribute like name.
3. **Cuisine:** Represents a culinary tradition (e.g., Italian, Thai) with an attribute like cuisine name (protected for access by subclass).
4. **FoodCategory:** Represents a food category (breakfast, lunch, dinner, or snack) with an attribute like category.
5. **Diet:** Represents a dietary restriction (e.g., vegan, vegetarian) with an attribute like name.
6. **FoodItem:** Represents a specific food item with attributes like name, ingredients, category (FoodCategory object), and diet (Diet object). It inherits cuisine_name from the Cuisine class.

2.2 Classes Involved:

1. **User:**
   - **Attributes:**
     - name (string)
     - * allergies (Allergen)

All member attributes are private.

   - **Methods:**
     - User(string n = “”, Allergen *a = nulltptr) is a both a default constructor and parameter constructor
     - ~User() is a destructor.
     - string getName() is an accessor of name
     - vector getAllergies() is an accessor of allergies
     - void setName(string n) is a mutator of name
     - void add_allergy(Allergen allergy)adds a new Allergy object to the allergy list.
     - void remove_allergy(string allergy_name) removes an Allergy object with the specified name.
     - void load_user_data(istream filename) reads user data from a local file
     - void save_user_data(ostream filename) writes user data to a local file
     - void edit_profile(string new_name, Allergen *a) allows editing of user information
2. **Cuisine:**
   - **Attributes:**
     - cuisine_name (string)

   Member function is protected, then it is derived class FoodItem can access the member attribute (cuisine_name).

   - **Methods:**
     - Cuisine(string n = “”) is both a default constructor and parameter constructor

       § string getCuisine() is an accessor of name

§ void setName(string n) is a mutator

3. **FoodCategory:**

    - **Attributes:**

        § category (string) This is food category (breakfast, lunch, dinner, or snack).

Attribute category is private.

- **Methods:**

    § FoodCategory(string c = “”) is both a default constructor

    § String getCategory() is an accessor of category.

    § void setCategory(string c) is a mutator of category.

4. **Allergen:**
    - **Attributes:**
        - name (string)

Attribute name is private.

- **Methods:**
    - Allergen(string n = “”) is both a default constructor and parameter constructor.
    - string getName() is an accessor of name
    - void setName(string n) is a mutator of name

5. **Diet:**
    - **Attributes:**
        - name (string)

Attribute name is private.

- **Methods:**
    - Diet(string n = “”) is both a default constructor and parameter constructor.
    - string getName() is an accessor of name
    - void setName(string n) is a mutator of name

6. **FoodItem (Class inheriting from Cuisine):**
    - **Attributes:** (Inherits cuisine_name from Cuisine)

        § name (string)

        - * ingredients (string)
        - * category (FoodCategory)
        - diet (Diet)

All member attributes are private.

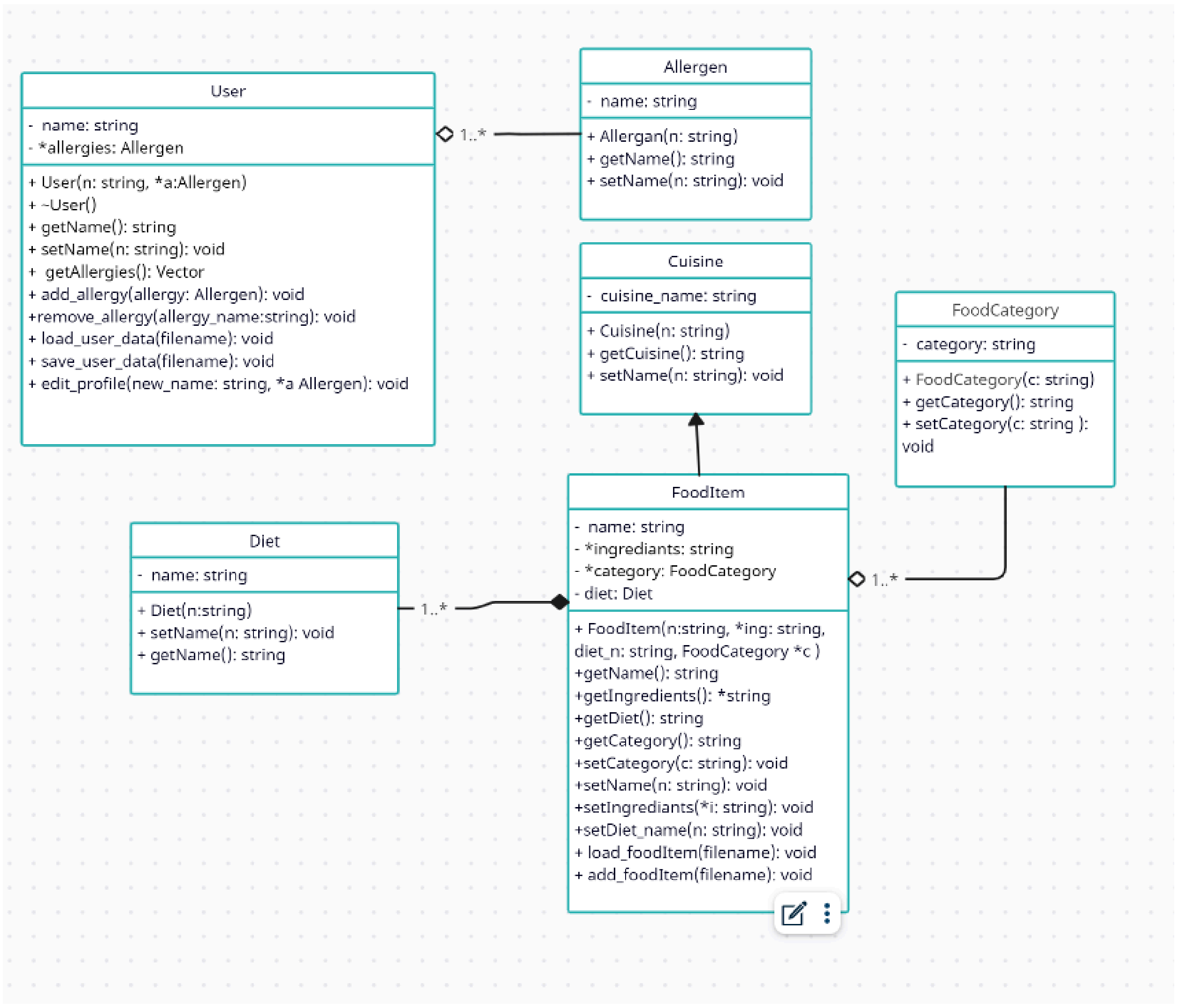
   - **Methods:**
        - FoodItem(string n= “”, string *ing = nullptr ,string diet_n = “”, FoodCategory *c) is both a default constructor and parameter constructor.
        - string getName() is an accessor of name.
        - string *getIngredients() is an accessor of ingredients
        - string getDiet() is an accessor of diet
        - string getCategory() is an accessor of category
        - void setCategory(string c) is a mutator of category
        - void setName(string n) is a mutator of name
        - void setIngredients(string *i) is a mutator of ingredients
        - void setDiet_name(string n) is a mutator of diet name
        - void load_foodItem(istream filename) reads food item data from a local file (e.g., food_data.txt)
        - void add_foodItem() add new food item to local file

<u>**2.3 Class Relationships**</u>:

- **Inheritance:**
  - FoodItem inherits from Cuisine. This establishes a parent-child relationship where each FoodItem is a type of Cuisine, this means that FoodItem will inherit cuisine_name attribute from Cuisine making it easier to split FoodItem into categories where the user will find it simpler to find FoodItem normally.
- **Aggregation:**
  - <u>User and Allergy</u>: This is an aggregation relationship, where User has a pointer to Allergen object. The Allergen pointer receives a list of Allergen objects created in int main, this means that the relationship between User and Allergen is a “has-a” relationship. The Allergen objects still exist even if a User object is destroyed. This is used to associate a list of allergens a user is allergic to.
  - <u>FoodItem and FoodCategory</u>: This is an aggregation relationship, where FoodItem has a pointer to FoodCategory object. The FoodCategory pointer receives an object of FoodCategory created in int main, this means that the relationship between FoodItem and FoodCategory is a “has-a” relationship. This is used to categorize food items to breakfast, lunch, dinner, and snack.
- **Composition**
  - FoodItem and Diet**:** This is a composition relationship. This means that a Diet cannot exist without a FoodItem, if a FoodItem is destroyed diet will be destroyed too. This too is done to enhance the experience of the user while searching for the FoodItem.

3.0 Section C

3.1 UML:-



User
- name: string
- *allergies: Allergen
+ User(n: string, *a:Allergen)
+ ~User()
+ getName(): string
+ setName(n: string): void
+ getAllergies(): Vector
+ add_allergy(allergy: Allergen): void
+remove_allergy(allergy_name:string): void
+ load_user_data(filename): void
+ save_user_data(filename): void
+ edit_profile(new_name: string, *a Allergen): void
1..*
Allergen
- name: string
+ Allergan(n: string)
+ getName(): string
+ setName(n: string): void
Cuisine
- cuisine_name: string
+ Cuisine(n: string)
+ getCuisine(): string
+ setName(n: string): void
FoodCategory
- category: string
+ FoodCategory(c: string)
+ getCategory(): string
+ setCategory(c: string): void
Diet
- name: string
+ Diet(n:string)
+ setName(n: string): void
+ getName(): string
1..*
FoodItem
- name: string
- *ingrediants: string
- *category: FoodCategory
- diet: Diet
+ FoodItem(n:string, *ing: string, diet_n: string, FoodCategory *c)
+getName(): string
+getIngredients(): *string
+getDiet(): string
+getCategory(): string
+setCategory(c: string): void
+setName(n: string): void
+setIngrediants(*i: string): void
+setDiet_name(n: string): void
+ load_foodItem(filename): void
+ add_foodItem(filename): void
1..*