Personal Dietary Manager Application:

Software Requirement Specifications

1. Introduction

1.1. Purpose

1.1.1. The purpose of this document is to present the software requirement specifications (SRS) of our Personal Dietary Manager Application. This application is being designed, developed, and tested in accordance with the requirements of COMP 5541: Tools and Techniques for Software Engineering, at Concordia University for the Winter 2020 semester.

1.1.2. The intended audience of this SRS are the COMP 5541 professor, the marking teaching assistant, and the members of our group who are developing this software application.

1.2. Scope

1.2.1. The software product to be produced is a dietary manager application. The name of this application is the “Personal Dietary Manager.”

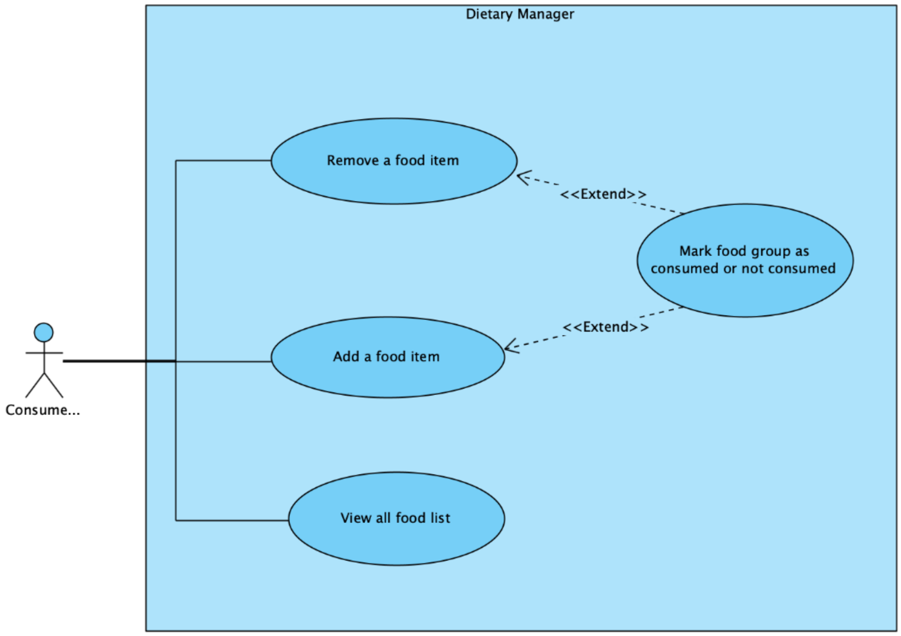
1.2.2. The objectives and goals are of this software project are:

1.2.2.1. To satisfy the requirements of COMP 5541
1.2.2.2. To deliver an application the is user-friendly, correct, maintainable, portable, and high-quality
1.2.2.3. To deliver an application that improves diet tracking
1.2.2.4. To work well as a team, using tools use as IDEs, version control (e.g. GitHub), UML software (e.g. Visual Paradigm), Java libraries (e.g. JavaFX), and Java Unit Testing
1.2.2.5. To present each iteration of our software project during lab sessions

1.2.3. There are three deliverable iterations of this software. The following are the functional requirements of each iteration:

1.2.3.1. The first iteration will be able to:

1.2.3.1.1. Graphical view of use cases:



1.2.3.1.2. Show a list to the user of all foods which they have added
1.2.3.1.3. Allow the user to add in a food item into a list of food
1.2.3.1.4. Allow the user to remove a food item from their food list
1.2.3.1.5. Mark none, some, or all the four food groups as eaten or not eaten

1.2.3.2. The second iteration will be able to:

1.2.3.2.1. Sort the list of food items that the user views based on food item attributes (e.g. serving, time, and food group)
1.2.3.2.2. Allow the user to select if food in food list is consumed or not consumed
1.2.3.2.3. Allow the user to hide and unhide food items based on if food item consumed or not
1.2.3.2.4. Create 2 mutually exclusive food attributes called indining and outdining
1.2.3.2.5. Show the displayed total nutrition elements (e.g. total calories, sodium) of the foods in the current displayed list; in a separate list, show these total nutrition elements of all consumed foods

1.2.3.3. The third iteration will be able to:

1.2.3.3.1. Store some or all food items in the food list of the Personal Dietary Manager in a database. This database will be stored on the disk. This database allows for the reuse of food items after the application has been closed

1.2.3.3.2. Implement a class that allows for opening and closing a database connection

1.2.3.3.3. Allow for start-up diets to be loaded from the database into the Personal Dietary Manager.

1.3. Definitions, acronyms, and abbreviations

1.3.1.1. Indining: A food attribute of a food item that was consumed at the home of the user

1.3.1.2. Foot attribute: A characteristic of a food item (e.g. serving size, calories, indining, time, retailer)

1.3.1.3. Food Item: A single instance of a food that is characterized by food attributes

1.3.1.4. Food Group: A food attribute that describes which food group the food item belongs to (e.g. fruits/vegetables, meat/alternatives, milk/alternatives, or grains)

1.3.1.5. Meal: A food attribute that describes if the meal is considered breakfast, lunch, dinner, or a snack

1.3.1.6. Outdining: A food attribute of a food item that was consumed outside of the home of the user

1.3.1.7. Retailer: A food attribute of the restaurant, grocery store, or name of the business where the food item was purchased (e.g. IGA, Wendy’s)

1.3.1.8. Serving: A food attribute that describes the amount and nutrition of the food item consumed (e.g. 30 ml, 250 g, ½ cup; calories, fat, sodium, sugar)

1.3.1.9. Type: A food attribute that describes user-specified additional information about the food item (e.g. homemade, bought)

1.3.1.10. User: The person who operates and directly interacts with the product

1.4. References

1.4.1. N. Houari, “COMP 5541: Project, Winter 2020”

1.4.2. IEEE Computer Society, “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Std 830-1998, Oct 1998

1.5. Overview

1.5.1. The remainder of this SRS further describes functions of the Personal Dietary Manager.

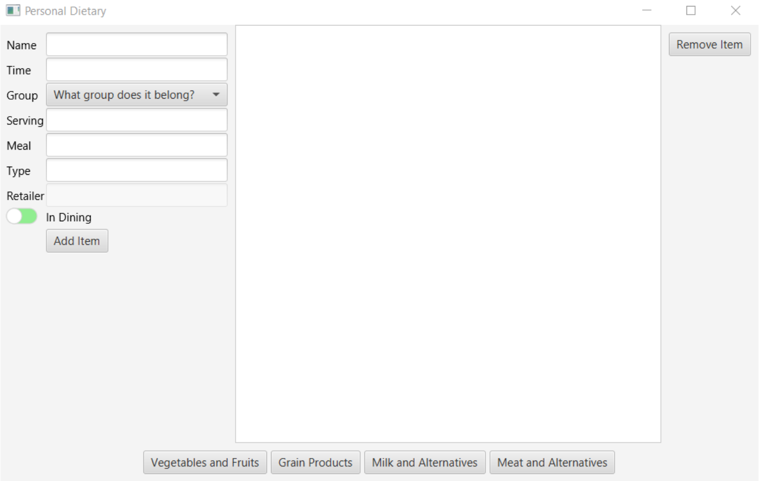
2. Overall Description

2.1. Product Perspective

2.1.1. The Personal Dietary Manager is a standalone product that is independent of other products and is self-contained

2.1.2. The user interface for increment 1 has the following characteristics:

2.1.2.1. A user interface like this:



2.1.2.2. A full-screen windows that can be resized
2.1.2.3. Buttons that allow a food item to be added and deleted
2.1.2.4. Text fields that allow for entering food attributes (name, time, serving, meal, type, and retailer)
2.1.2.5. A list of food items that is always displayed
2.1.2.6. Buttons that indicate whether a food group has been consumed or not

2.1.3. The user interface for increment 2, in addition to everything in increment 1, has the following characteristics:

2.1.3.1. A button to hide or unhide consumed food items
2.1.3.2. Additional text fields to add in date, serving amount, calories, fat, sodium, and sugar
2.1.3.3. 2 lists that display total nutritional information. The first list displays total nutrition of the food items currently displayed. The second list displays total nutrition of the consumed food items.

2.1.4. The user interface for increment 3, in addition to everything in increment 2, has the following characteristics:

2.1.4.1. Allow the user to save a food item to a database
2.1.4.2. Allow the user to retrieve an item from the database
2.1.4.3. Allow the user to see all food items in the datbase

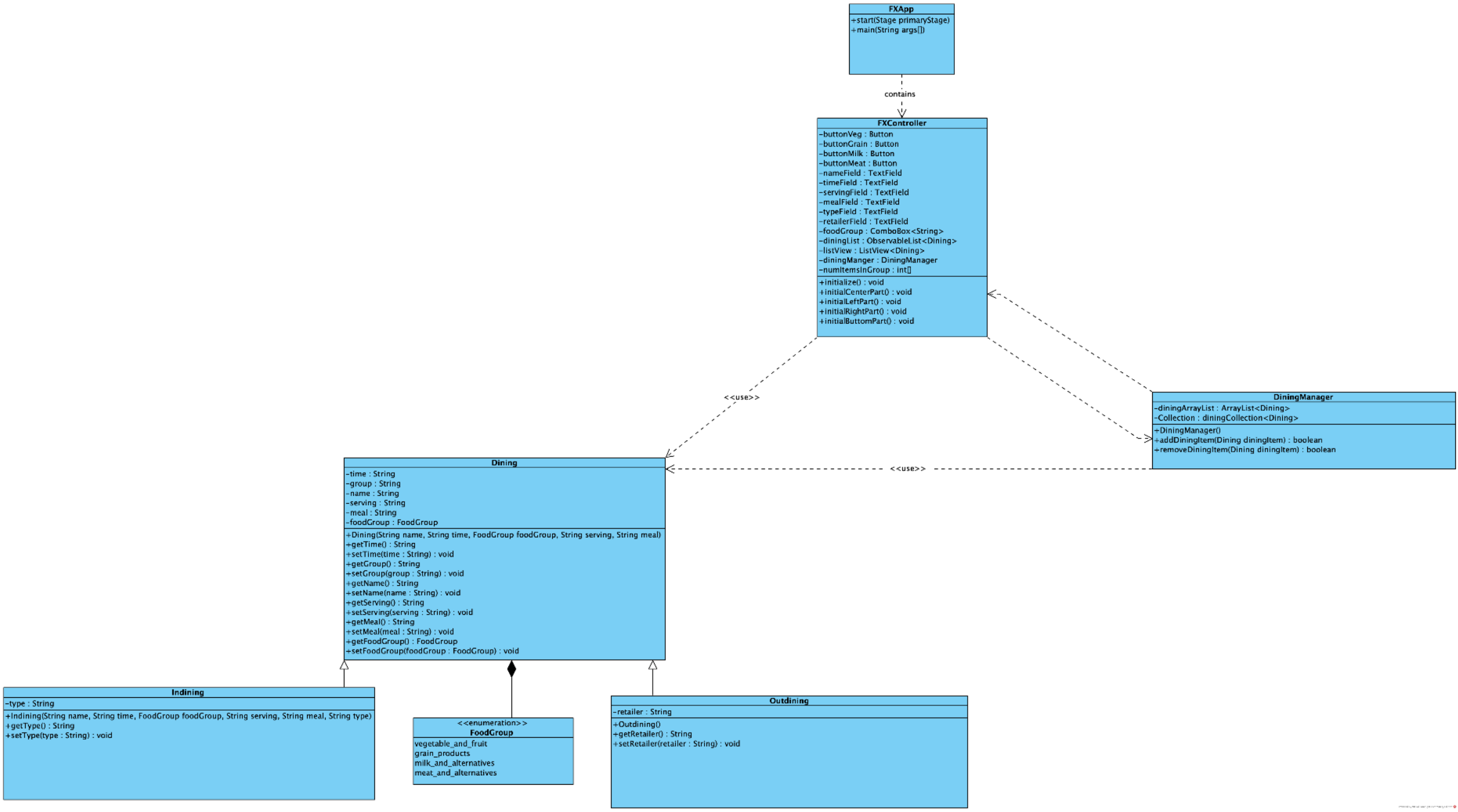
2.1.5. The following software products will be used in developing this application:

2.1.5.1. Windows or Mac OS
2.1.5.2. IntelliJ IDE
2.1.5.3. JAVA SDK
2.1.5.4. JAVA FX library
2.1.5.5. Visual Paradigm
2.1.5.6. GitHub

2.2. Product Functions / Use cases

2.2.1. See section *1.2.3*

2.3. UML Class Diagram


FXApp
+start(Stage primaryStage)
+main(String args[])
contains
FXController
-buttonVeg : Button
-buttonGrain : Button
-buttonMilk : Button
-buttonMeat : Button
-nameField : TextField
-timeField : TextField
-servingField : TextField
-mealField : TextField
-typeField : TextField
-retailerField : TextField
-foodGroup : ComboBox<String>
-diningList : ObservableList<Dining>
-listView : ListView<Dining>
-diningManger : DiningManager
-numItemsInGroup : int[]
+initialize() : void
+initialCenterPart() : void
+initialLeftPart() : void
+initialRightPart() : void
+initialButtomPart() : void
<<use>>
DiningManager
-diningArrayList : ArrayList<Dining>
-Collection : diningCollection<Dining>
+DiningManager()
+addDiningItem(Dining diningItem) : boolean
+removeDiningItem(Dining diningItem) : boolean
<<use>>
Dining
-time : String
-group : String
-name : String
-serving : String
-meal : String
-foodGroup : FoodGroup
+Dining(String name, String time, FoodGroup foodGroup, String serving, String meal)
+getTime() : String
+setTime(time : String) : void
+getGroup() : String
+setGroup(group : String) : void
+getName() : String
+setName(name : String) : void
+getServing() : String
+setServing(serving : String) : void
+getMeal() : String
+setMeal(meal : String) : void
+getFoodGroup() : FoodGroup
+setFoodGroup(foodGroup : FoodGroup) : void
Indining
-type : String
+Indining(String name, String time, FoodGroup foodGroup, String serving, String meal, String type)
+getType() : String
+setType(type : String) : void
<<enumeration>>
FoodGroup
vegetable_and_fruit
grain_products
milk_and_alternatives
meat_and_alternatives
Outdining
-retailer : String
+Outdining()
+getRetailer() : String
+setRetailer(retailer : String) : void