DESIGN DOCUMENT

for

Personal Dietary Application

Version 1.0

by Craig Boucher
Md Tanveer Alamgir
Fan
Osman
Xin

Contents

1 Introduction

The project undertaken in this COMP 5541 course involves creating an application that keeps track of dietary records for the user. This diet app has been designed to use the border pane layout for the main window.

This design document will provide details for the type of software architecture used to develop the software and explain the design for the user interface. The architectural component illustrates the abstraction of the software classes involved and how they relate to each other to maniuplate and process data that the user interacts with. The interface design section will assess the process for the states the user goes through to interact with the personal dietary application.

1.1 Purpose

This document will serve as an illustration for the architectural design choices as well as an explanation for the properties of the user interface implemented for the Personal Dietary Application software. This software project is being completed for the COMP 5541 graduate diploma course at Concordia University. There will be diagrams to chowcase the type of the architecture used and a class diagram. Screenshots are used to provide a valuable perspective on the user interface. These graphical aids are described in further detail to demonstrate the functionality of the interface.

1.2 Scope

In order to provide proper design documentation for the Personal Dietary Application this document will be properly formatted and included all necessary information. The team responsible for surveying this document and creating the software will be able to use the robust explanation of the architectural model presented in this document to carry out the necessary work. The visual graphics demonstrating the user interface design choices will serve as a guide for the team to orchestrate the proper performance of the software.

1.3 Definitions and Abbreviations

1.3.1 Definitions

Term	Definition
Model View Controller	Software architecture that renders funcionality between three components. The view is the user interface. The model stores the data and the controller mediates data transfer between the view and model.
Date	Allows user to enter day and month of an entry for an item.
Consumed	The user will be able to mark a food item as consumed (eaten) or not.

1.3.2 Abbreviations

Abbreviation	Term
MVC	Model View Controller
GUI	Graphical User Interface
UML	Unified Modeling Language
PDA	Personal Dietary Application

1.4 References

https://upload.wikimedia.org/wikipedia/commons/a/a0/MVC-Process.svg
http://users.encs.concordia.ca/ paquet/wiki/images/e/ee/Phase2final.pdf

1.5 Overview

The rest of the document is composed of two main sections. First, there is a section dedication to Architectural Design. Lastly, the largest section, consists of Software Interface Design. In the portion concerning the software architecture the three smaller sections are related to the reasoning for choosing the architecture, a diagram illustrating how the architecture behaves abstractly, and the final section details how software files will operate together on the computer.
The interface section will provide documentation regarding the system, modules, and dynamic interfaces for the software project.

2 Architectural Design

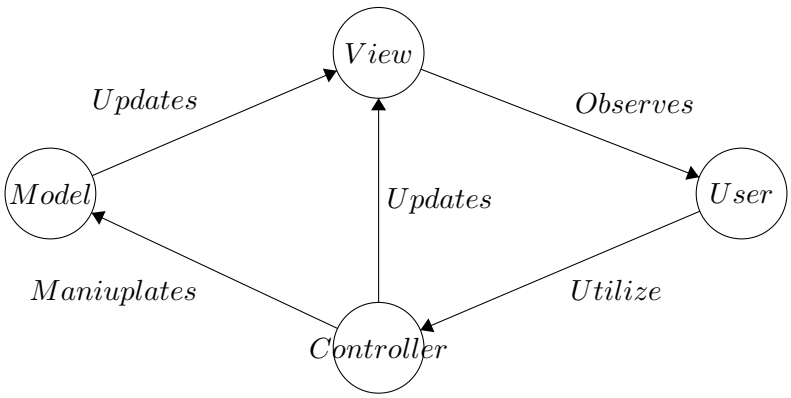
2.1 Rationale

The Model View Controller (MVC) architecture design pattern has been chosen for this software project. Model, View, and Controller are the three components which comprise this architecture.

The purpose of the Controller model is to navigate and facilitate the transfer of data between the View and the Model. The Controller is responsible for acquiring data input from the user, manipulate the model with the necessary memory changes, and potentially provide updates to the View if and when exceptions or errors are thrown.

The View is where the (graphical) user interface rests. The information stored in the model is used by the View to display everything necessary to the user. Everything the user interacts with rests with the View, from input functionality to button objects. Whenever data is manipulated in the Model the View will reflect this accordingly.

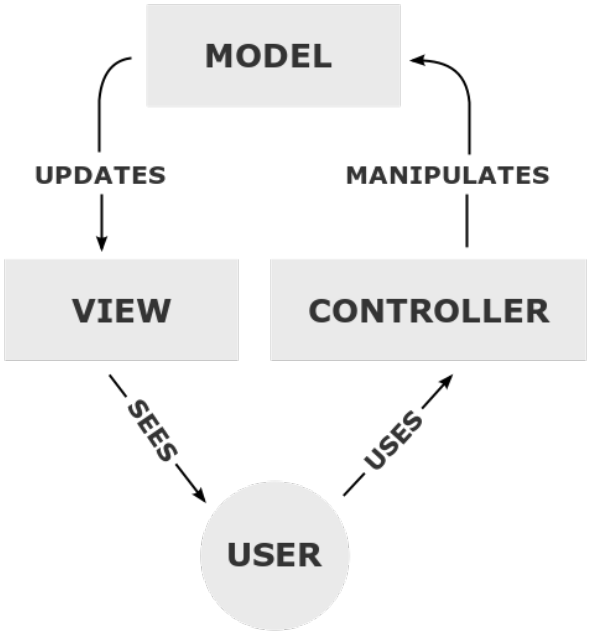
The objective of the Model is to store all of the necessary information and data that the application requires to operate successfully. All of the data is stored in local memory. Every food item and all of the details of each item is stored in the Model module. There is a list of every food item added by the user that is stored in the Model, and whenever a change is made (adding or removing food items), the View will graphically respond to these changes.



The software specifications outlined in the project instructions specifically mandated the MVC architecture. The purpose of this architecture is to provide a relevant abstrac-

tion of how desktop, mobile, or web applications operate when requiring constant use by a user. The abstraction allows three distinct modules to operate independently of each other and be capable of being updated separately. Different developers can be assigned to develop for the various modules with minimal cooperation required between them as the core functionality of each module remains the same across updates.

2.2 Software Architecture Diagram



2.3 System Topology

3 Software Interface Design

3.1 System Interface Diagrams

Our personal dietary application has only system level interface. The application does not employ any software or hardware interfaces. GUI is the system level interface. GUI allows the user to interact with the application. Using GUI user will be able to add a food item, mark it as eaten or not eaten, hide an added item, remove an item and the update of the dietary.

3.1.1 User Interface

User will interact with the computer using the user interface. We tried to make it as user friendly as possible by placing the components with an order of their importance and utilizing the full screen mode. Below are the few points we took into consideration for user interface:

- User Friendly: Making it user friendly by putting important components according to user's view point.
- Easy to find information: Since the application is all about their dietary, so it was important to make it easy to calculate and display their consumed and need to consumed items. We make it as handy as possible.
- Guiding user: Guiding user by displaying different colour if they miss out something while inputting a food item.

3.2 Dynamic Models of System Interface

The system interfaces that establish the interaction between user and computer are described below:

3.2.1 Landing page

Below is the first screen user will see when they will launch the application. Here is the description of every options available to user:

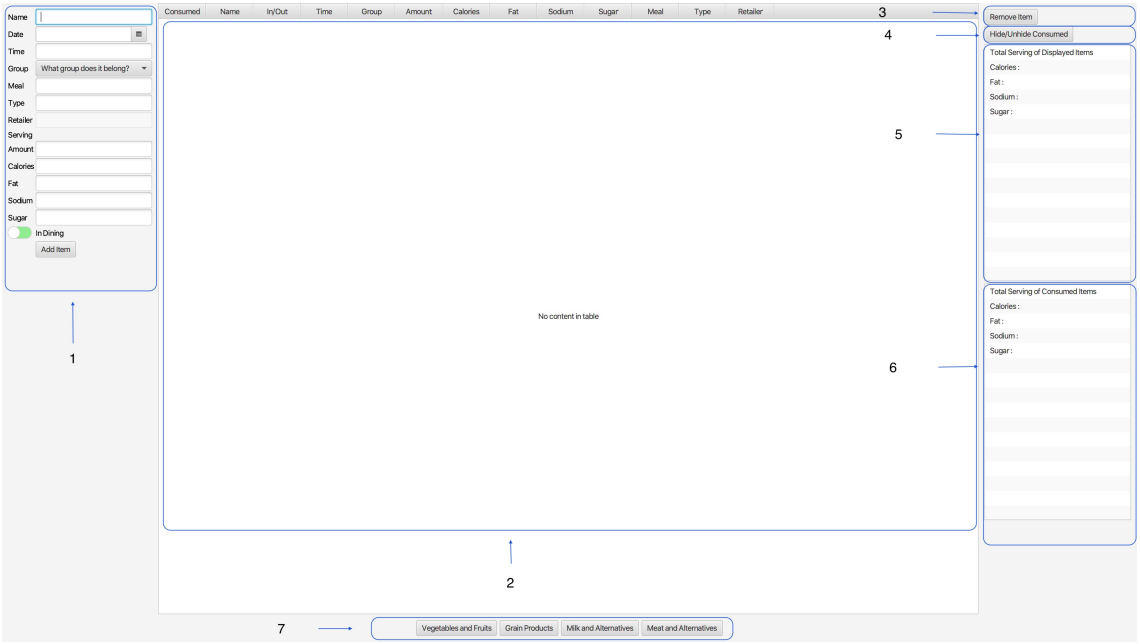


Figure 3.1: Landing page

1. Add a food item: User can input the food details to have it added.
2. Display the added food: It's a list of all food added by user
3. Remove a food item: It's a button to remove a food item from added list.
4. Hide/unhide Consumed food: This button gives the user ability to hide/unhide consumed food.
5. Total serving: It will show the total serving of all the items in the added food list.
6. Serving of consumed item: This will only show the total serving of consumed item(s).
7. Food group: Based on the food item(s) added and consumed these 4 button will indicate which group they belong to.

3.2.2 Add Food Item Scenario

Below gives the ability to an user to add a food item by inserting the details of the food:

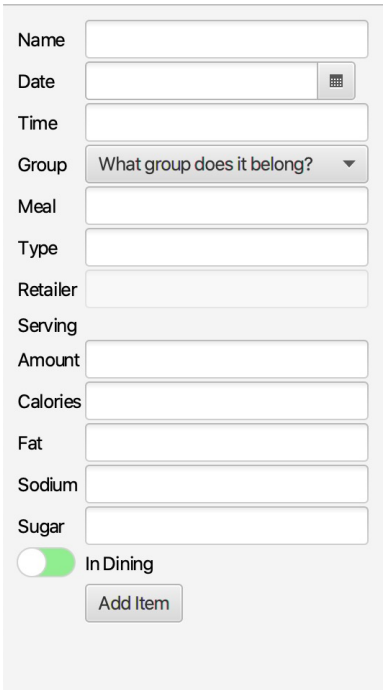


Figure 3.2: Add a food item

1. Name: User will insert the name of the food.
2. Date: User will select the date from the calendar on the right side of the text box.
3. Time: User will insert the time of the food consumed. Time the is in 24h formate.
4. Food Group: User will select the respective food group from the drop down menu.
5. Meal: User will insert the meal they consumed the food at.
6. Type: This field is for user to indicate whether the food was bought, homemade. If the food was consumed at out dining then this field will get greyed out.
7. Retailer: This is to identify the name of the retailer if the food was consumed from out dining. If in dining is selected using the toggle switch at the bottom then this field will be greyed out.
8. Amount: The amount of food consumed by the user.
9. Calories: The amount of calories presented in the consumed food.
10. Fat: The amount of fat presented in the consumed food.
11. Sodium: The amount of Sodium presented in the consumed food.

12. Sugar: The amount of Sugar presented in the consumed food.

13. Toggle Switch (In dining/Out Dining): This toggle switch will allow the user to choose whether the food was consumed inside of outside.

14. Add Item: After Filling out all the details of the food user will press the Add Item button to add the food item.

15. Validation: If user does not fill out all the fields then the placeholder will show up to indicate which field needs to be filled out and what to put inside that field

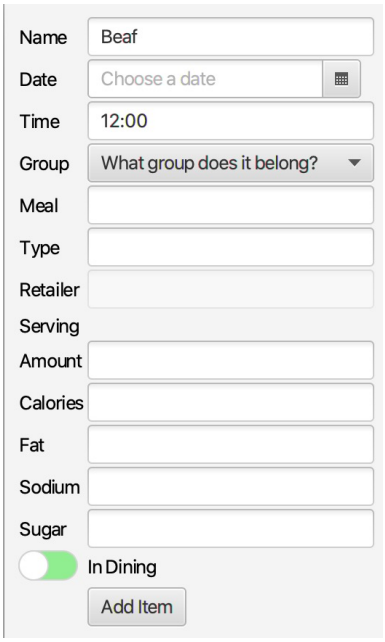


Figure 3.3: Validation of input

3.2.3 Remove Food Item Scenario

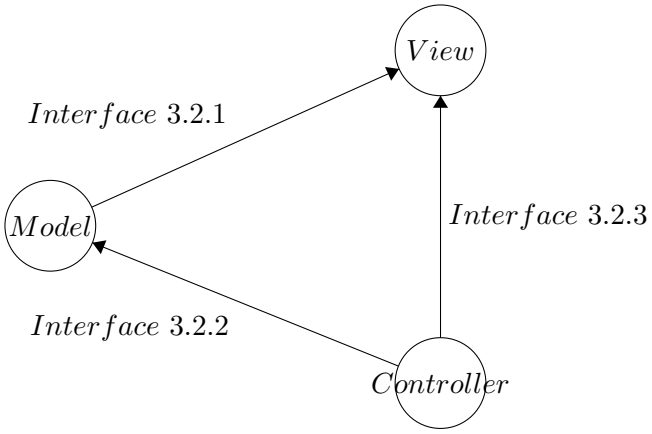
3.2.4 Set Food Item as Consumed Scenario

3.2.5 Set Food Item as Unconsumed Scenario

3.2.6 Hide Consumed Diet Scenario

3.2.7 Unhide Consumed Diet Scenario

3.3 Module Interface Diagrams



3.3.1 View Interface

FXApp

FXController

DiningTableRow

3.3.2 Model Interface

3.3.3 Controller Interface